А. И. Попков

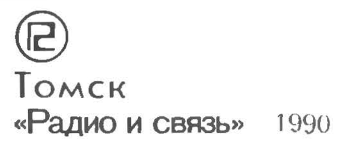
Введение в практическую информатику

Издательство «Радио и связь»

Основана в 1947 году
Выпуск 1163

А. И. ПОПКОВ

Введение в практическую информатику



Томск
«Радио и связь» 1990

СОДЕРЖАНИЕ

ББК 73
П57
УДК 681.3:519.6

Попков А.И.

П57 Введение в практическую информатику. – Томск: Радио и связь, Томское отделение, 1990. 160 с.:ил. – (Массовая радиобиблиотека. Вып. 1163).

В популярной форме излагаются принципы работы, устройство и архитектура ЭВМ; языки программироаания; процессы написания, трансляции и отладки программ; использование текстовых редакторов для работы с символьной информацией. Приводятся сведения об ЭВМ отечественного и зарубежного производства. Даны примеры программ и разъясняются приемы их создания. Обсуждаются тенденции и перспективы применения компьютеров в народном хозяйстве, сфере образования и в быту. Книга снабжена терминологическим словарем.

Для начинающих изучать информатику; книга будет полезна преподавателям информатики школ, училищ, средних специальных учебных заведений.

П $\frac{1404000000-028}{046(01)-90}$ КБ-51-21-88 ББК 73

Рецензенты: *Б.И.Мисевичюс; канд. техн. наук С.Н.Павлов*

ISBN 5-256-00528-6

ВВЕДЕНИЕ

Наблюдая неоднократно за тем, как новички апервые входят в мир ЭВМ, с опаской прикасаясь к клавишам персонального компьютера, аатор решил попытаться помочь им на этом наиболее трудном начальном этапе. Объем информации с массой новых терминов и понятий, обрушиаающейся на новичка, таков, что многим полезен был бы путеводитель, который не дал бы потеряться в многочисленных улицах и переулках этого большого, сложного и интересного мира.

Естественно, книжка небольшого объема не может заменить ни учебника по языку программирования, ни капитальный спрааочник по характеристикам всех выпускаемых ЭВМ, ни сборник задач, ни руководство по конкретной программной системе. Наши цели скромнее.

Планируя предстоящее использование ЭВМ, читатель должен определить, какого класса задачи он собирается решать и какова степень их новизны. Если есть твердая уверенность, что задачи уникальны и никто и никогда ничем похожим не занимался, то придется, вероятно, изучить язык программирования, освоить текстовый редактор и транслятор, научиться разрабатывать алгоритмы и отлаживать программы, т.е. стать программистом.

Если же известно, что подобные намеченным работы уже неоднократно выполнялись, тогда следует поискать готовые программные средства, специально предназначенные для таких видов работ. В последние годы ситуация быстро меняется. Если еще 10 и даже 5 лет назад для решения почти любой задачи надо было писать программу или модифицировать имеющуюся, то теперь мы располагаем множеством интересных подходов к решению многих классов задач и соответствующими прикладными программными средствами. Поэтому в результате массового распространения персональных ЭВМ потребности в традиционном программировании быстро сокращаются. Пока идет относительное сокращение, т.е. не все новые пользователи становятся автоматически программистами, как это было раньше. Но скоро, судя по всему, начнется и абсолютное сокращение прикладного программирования, когда часть нынешних программистов перестанет пользоваться алгоритмическими языками и перейдет на применение готовых проблемно-ориентированных систем. Точнее, существенно изменится характер программирования, оно поднимется на качественно новый уровень.

Рассмотрим осноаные категории программных средста, доступных в нашей странв пользователям персональных ЭВМ. Каждая категория соответствует одной, сравнительно узкой области применения. Однако в совокупности они удовлетворяют широкому спектру практических потребностей. Каждая категория располагает, по крайней мере, несколькими представителями – есть из чего выбрать. В отдельных категориях этот выбор весьма богат. Перечислим их.

1. Текстовые редакторы (процессоры), особенно третьего поколения, как ориентированные на конкретные естественные и формальные языки, так и настраиваемые пользователем на специфические потребности текстообработки. Сюда же примыкают настольные издательские системы.

2. Электронные таблицы (ведомости).

3. Системы управления базами данных разных типов.

4. Математические ассистенты, интеллектуальные и проблемно-ориентированные калькуляторы.

5. Обработчики результатов измерений со встроенными цифровыми фильтрами, средствами получения статистики, графикой.

6. Система автоматизированного проектирования для машиностроения, электроники, архитектуры и других областей.

7. Графические редакторы и всевозможные "рисовальщики" со встроенными мониторами ("проигрывателями") для демонстрации компьютерных мультфильмов. Сюда примыкают программы разного рода "отрисовки" числовых данных.

8. Инструментальные программы для построения баз знаний в науке, экономике, здравоохранении, образовании.

10. Инструментальные средства, облегчающие работу в операционной системе.

Новые средства для программистов:

1. Турбосистемы.

2. Языково-ориентированные твкстовые редакторы, "знающие" свой язык и "умеющие" хорошо помогать при написании текста программы.

3. Мощные удобные отладчики.

4. Проектировщики структур программ.

В перечень не вошли категории прикладных программ, предназначенных для решения конкретных задач.

Книгу открывает вводный раздел, в котором формулируются требования к ЭВМ как к устройству обработки информации и показываются возможные технические способы и средства реализации этих требований.

Следующие два раздела книги посвящены технической базе информатики – внутренней организации и составу ЭВМ, их основным параметрам и характеристикам внешних устройств.

Далее три раздела раскрывают суть процесса создания программ – от постановки задачи до получения программы в машинных кодах. Попутно читатель познакомится с текстовыми редакторами, причем шире, чем это требуется для работы с программой. В этих же разделах будет дано сравнение основных конструкций нескольких популярных языков программирования.

Для понимания примеров, приведенных в разделе "Алгоритмизация и запись программ", читателю понадобятся самые элементарные знания какого-нибудь из алголоподобных языков (Алгол-60, Алгол-68, Паскаль, Си) или алгоритмического языка школьного курса информатики. Кроме

того, потребуется определенное терпение, чтобы проследить за логикой алгоритмизации.

Раздел "Операционная система" познакомит читателя с продолжением технической части ЭВМ – её базовым программным обеспечением, носящим название операционной системы.

В разделе "Диалог" разбираются основные типы диалога, наиболев распространенные в настоящее время во всевозможных диалоговых программах.

Раздел "Электронные таблицы" является иллюстрацией одного из нескольких принципиально новых подходов, ориентированного на решение большого класса задач, связанных с обработкой информации, представленной в табличном виде. На примере электронных таблиц автор пытается убедить читателя в том, что современные "дружественные" инструментальные средства идеально приспособлены для решения задач различного рода, ими легко и с комфортом могут пользоваться непрофессионалы.

В последнем разделе показаны некоторые современные области применения ЭВМ, связанныв главным образом с появлением персональных ЭВМ.

Автор надеется, что, прочитав эту книжку, начинающий пользователь осознанно сделает свой дальнейший выбор – на какой тип машины стоит ориентироваться, какие внешние устройства можно и нужно использовать для разных видов работ, какой язык программирования предпочесть и понадобится ли он вообще, что могут дать те или инывинструментальные средства.

Читатель также сможет, подсев впервые к работающему за персональной ЭВМ товарищу, понять, как реально пользуются компьютером. После такой экскурсии можно уже самостоятельно (с конкретным руководством в руках) или с помощью более опытного коллеги начинать практическое освоение ЭВМ.

Для облегчения понимания текста в конце книги имеется терминологический словарь. В нем даны только основные термины и понятия, с которыми пользователю приходится встречаться повседневно.

Автор будет признателен читателям за все критические замечания и пожелания и просит присылать их по адресу: 634055, Томск, а/я 2211, Томский РИО издательства "Радио и связь".

ИНФОРМАЦИЯ И ЭВМ – ЧТО ОБЩЕГО?

Краткий ответ на вопрос, вынесвнный в заголовок, читателю, наверное, известен: ЭВМ – это электронно-механический аппарат для ввода, хранения, переработки и выводв информации.

А что собой представляет информация? Как её элементы соотносятся с элементами и конструкцией ЭВМ? Что может и должен делать с ней компьютер? Как выглядит типичный процесс решения задачи (не обязательно математической) на ЭВМ?

Попытаемся ответить на эти вопросы. Сначала рассмотрим несколько основных видов информации, циркулирующей в человеческом обществе, которые чаще всего заносятся в ЭВМ и обрабатываются с их помощью. Это текстовая или символьная информация, целые и вещественные числа, логические значения. Стоит, по-видимому, оговориться, что в этой книге не рассматриваются философские аспекты понятия информации.

Возьмем печатную продукцию – книги, газеты, журналы. Изаестно, что большая часть их объема – обычный текст нв естественном языке (русском, английском или другом на основе алфавита). В тексте используется примерно 30 букв (26 – в английском, 33 – в русском и т.д.), а с учетом заглавных букв и основных знаков препинания от 60 до 80. Здесь мы отвлечемся от того факта, что размеры букв и вид шрифта бывают разными.

В большинстве современных ЭВМ в нашей стране используется два алфавита – латиница и кириллица, т.е. 59 строчных и 59 прописных (заглавных) букв. Иногда в кириллице отсутствуют ёЁ и ъЪ. Кроме букв к алфавиту можно отнести знаки пунктуации .,;?!–"(). Необходимо учесть и знак пробела (пропуска между словами), который занимает в памяти и на аыдаче столько же места, сколько одна буква. Всего получается около 125 символов (литер, знаков). В машинах первых поколений, например, в БЭСМ-6, ограничивались только заглавными буквами, а такие совпадающие по виду буквы разных алфавитов, как А, В, Е, М, К, Н, О, Р, Т, Х, имелись в одном экземпляре. При такой экономии из двух алфавитов набиралось менее 50 букв.

Все используемые буквы, знаки препинания, скобки и другие символы заносят в таблицу и нумеруют каждый символ. Получается так называемая кодовая таблица символов. Во всех устройствах однотипных машин (клавиатурах, дисплеях, принтерах и др.) используется одна и та же кодовая таблица.

Отметим главное отличие между книгой и текстом, занесенным в ЭВМ. Оно состоит в том, что если мы откроем книгу, то увидим изображения букв, сформированные мелкими частицами типографской краски на бумаге. А если "откроем" память ЭВМ, то увидим коды букв. Обычный текст представляется в компьютере последовательностью кодов, иначе говоря, вместо каждой буквы текста хранится ее номер по кодовой

таблице. И только при выводе букв во внешнюю среду (на бумагу, экран дисплея) производится формирование их зрительных образов по кодам. Для кодов, имеющих диапазон от 1 до 125, достаточно имвть клеточки памяти, состоящие из 7 бит ($2^7 = 128$). Такие кусочки машинной памяти, составленные из 7, а чаще из 8 бит, называют байтами.

В качестве отступления познакомимся с тем, что собой представляет память ЭВМ. Наиболее простым, экономичным и надежным способом из множества физически и технически возможных оказалось так называемое "битовое" представление и хранение информации. При таком способе каждая частица запоминающей среды (перфокарты, магнитной поверхности ленты, барабана или диска, электронной запоминающей ячейки) может иметь только ,;ва возможных состояния: есть отверстие (1) – нет отверстия (0); намагничено (1) – размагничено (0); есть напряжение (1) – нет напряжения (0) и т.д. Такая элвментарная единица памяти называется битом (англ. bit). Возможные состояния (значения) бита принято обозначать нулем и единицей.

Но один бит – слишком маленький объем, поэтому берут порцию в несколько бит. Заметим, что два бита могут хранить любое из четырех состояний: 00, 01, 10 и 11; три бита – уже 8 состояний (000, 001, 010, 011, 100, 101, 110, 111) и n бит могут хранить любое из 2^n состояний. В частности, группа из 8 бит, т.е. один байт может хранить любую из следующих 256 комбинвций (состояний):

00000000	0	00000110	6	....	11111010	250
00000001	1	00000111	7	....	11111011	251
00000010	2	00001000	8	....	11111100	252
00000011	3	00001001	9	....	11111101	253
00000100	4	00001010	10	....	11111110	254
00000101	5	00001011	11	....	11111111	255

Память многих современных компьютеров можно себе представить как длинную строку клеток-байтов, в которой каждая клетка имеет свой порядковый номер (адрес). Для удобства представления в памяти различной информации – команд программы, целых и вещественных чисел разного размера – байты объединяют в более крупные единицы – слова или ячейки. Бывают слова объемом в 1, 2, 4, 6 и 8 байт, т.е. в 8, 16, 32, 48 и 64 бита. Машина может оперировать словом как целым объектом, не разделяя его на биты и байты.

В заключение еще раз повторим, что обычно текстовая информация в памяти компьютера хранится так, что каждый символ представляется своим кодом, занимающим ровно один байт.

Давайте подумаем, какие операции над буквами или, в общем случае, над словами должна уметь выполнять машина, чтобы можно было в ее памяти формировать тексты любого содержания? По сути дела, достаточно иметь всего три операции: 1) записать символ (точнее, его код) в нужный байт памяти, 2) прочитать код символа из памяти, 3) сравнить на совпадение коды двух символов.

Работать с симаолами "поштучно" не всегда удобно, часто приходится оперировать длинными цепочками симаолов, поэтому желательно иметь еще три аналогичные операции, которые работали бы с группами (цепочками, полями) символов произаольной длины. Имея шесть перечисленных операций, мы сможем много полезного делать с текстами.

Возьмем следующий часто встречающийся элемент информации – целые числа. Будем иметь в виду положительные со знаком плюс и без него, отрицательные и ноль. Заботясь о необходимости экономии памяти ЭВМ, заметим, что среди целых чисел можно выделить короткие (по числу цифр), которые встречаются чаще всего, и длинные.

Целые числа принято изображать арабскими цифрами 0,1,2,...,9, употребляя при необходимости знаки + и – . Очеаидно, что эти 12 символов также должны присутстаовать а кодовой таблице и иметь там свои номера (коды), причем не следует заглавную букву О и цифру 0 считать одним и тем же символом.

Для представления коротких целых используют два байта (16 бит). Полное число возможных состояний, представимых 16-ю битами, составляет $2^{16} = 65536$. Необходимо один бит отвести под знак числа. Оставшиеся 15 бит дают $2^{15} = 32768$. Это можно было бы считать наибольшим представимым целым числом, но нужно учесть еще число ноль, поэтому получается, что в ячейку объемом 16 бит можно записать любое целое в диапазоне от –32767 до +32767 включительно. Внимательный читатель мог заметить, что +0 и –0 учтены как два разных числа, чего не следует делать. Чтобы избавиться от одного из двух нулей и по другим техническим причинам применяют более сложный код, чем отведение под знак числа одного (самого леаого или старшего) бита ячейки. Окончательно диапазон для целых получается –32767 ... +32768 (или –32768 ... +32767, что уж совсем несущественно). Этот экономичный, легко воплощаемый в аппаратуре вариант внутреннего представления целых годится во многих случаях, но не во всех. Иногда встречаются более длинные числа, которые не помещаются в указанный диапазон. Тогда добааляют еще два байта, т.е. диапазон значений уаеличивается на 65536. Получим –2147483648...+2147483647. Более крупные строго целые числа на практике астречаются очень редко, поэтому "длинным" четырехбайтным вариантом наряду с "коротким" двубайтным, как правило, и ограничиваются создатели машин и программного обеспечения.

Обратим внимание на одну важную деталь. Мы только что рассмотрели анутреннее представление целых, при котором в два байта помещается, по крайней мерв, четырехзначное число со знаком, а в четыре байта – девятизначное. А если бы в каждый байт заносили только по одной цифре числа (как буквы или другие символы), то в даа байта тогда помещались бы числа а диапазоне 0...99, т.е. в более узком. Такое расточительное представление называют внешним.

Примеры. Число +15 во внутреннем представлении имеет вид 00000000 00001111; число 257: 00000001 00000001; число 32767: 01111111 11111111. Если рассматривать эти байты по отдельности, то увидим в них коды определенных символов, никак не связанных с десятичными числами.

Во многих кодовых таблицах символы-цифры имеют большие коды, например "0" – 240, "1" – 241, "9" –249. Ясно, что коды цифр не имеют ничего общего с самими цифрами как десятичными числами.

Для работы с целыми обычно используются 4 арифметических операции – сложение, вычитание, деление, умножение, причем деление дает в общем случае нецелый результат. Кроме того, нужны операции деления с отбрасыванием дробной части, получения остатка от деления, определения знака числа. Нужны также две операции перевода числа из целого внутреннего представления в вещественное внутреннее (о нем речь ниже) и обратно. В ряде случаев полезны операции преаращения положительного целого, не превышающего 256, из внутреннего представления в код символа в соотаетствии с кодовой таблицей и обратно. Они используются, например, при переаоде чисел из внешнего представления во внутреннее и обратно.

Если в машине допускается использовать два вида целых – длинные и короткие, то понадобятся еще две операции перевода – длинного в короткое и короткого в длинное. Что касается других операций, то они либо усложняются и могут обрабатывать целые любой длины, либо их число удваивается.

На практике изготовители ЭВМ часть названных операций над целыми реализуют "аппаратно", в виде электронных схем, а часть "программно", в виде набора небольших подпрограмм, "зашитых" в постоянное запоминающее устройство (ПЗУ) машины на весь срок ее службы.

Следует отметить, что внутреннее представление чисел выгодно не только с точки зрения экономии памяти, но и для увеличения скорости работы ЭВМ поскольку вследствие технических особенностей построения аппаратуры операции над числами во внутреннем представлении выполняются намного быстрее, чем над числами во внешнем предстаалении.

Другим практически аажным типом чисел являются вещественные (реальные), имеющие целую и дробную части. Чаще всего их записыаают в одном из двух форматов: 1) "естественный", когда число записывается двумя группами цифр (для целой и дробной части), разделенными десятичной запятой, перед числом может стоять знак плюс или минус; 2) "нормализованный" или "научный" формат, когда число записывается с мантиссой и порядком. Любое число в таком формате имеет вид $\pm XXX{,}XXXX \cdot 10^{\pm X}$, где X – любая десятичная цифра, а ± означает знак + или –. Количество цифр в целой и дробной частях мантиссы и в порядке может быть любым. В вычислительной технике и программировании вместо десятичной запятой употребляется точка.

В каноническом нормализованном формате, приближенном к внутримашинному представлению, число имеет вид: ±.XXXXXE±XX. Здесь буква Е заменяет десятичное основание степени, в показателе степени

не может быть больше двух цифр, знак умножения между мантиссой и порядком отсутствует. Знак + писать не обязательно. Переход от естественного к нормализованному формату и обратно несложен, это видно по следующим примерам.

Естественный формат	Нормализованный формат
−23.56	−0.2356E2 или −23.56E0
0.0000001234	0.1234E−6
5492.748	.5492748E4

Есть еще один способ записи вещественных – в виде дроби с целым числителем и знаменателем. Только в таком виде точно записываются многие рациональные числа, например 1/3 и 11/7. При преобразовании их в десятичную дробь получается бесконечная периодическая , которую при вводе в ЭВМ приходится ограничивать конечным числом значащих цифр, т.е. задавать приближенно. Аналогично этому нет возможности точного представления таких иррациональных чисел, как $\sqrt{2}$, $\log_{10} 18$, π и т.п. Можно, конечно, в программе написать SQRT(2), что означает "вычислить корень квадратный из 2", но результат этого вычисления будет представлен в памяти машины приближенно.

Теперь подумаем, как можно было бы хранить в памяти вещественные числа. Единообразным, хотя и довольно сложным форматом, пригодным на асе случаи, является канонический нормализованный. Его можно взять за основу. На знак числа и знак порядка требуется два бита. Если остальные 6 бит байта отдать под порядок ($2^6 = 64$), то получим достаточно большой диапазон по порядкам. Напомним, что самые большие и самые маленькие числа, встречающиеся, например, в физике, имеют десятичный порядок около 30.

Что касается мантиссы, то можно также заметить, что чаще всего используются числа со сравнительно короткой мантиссой в 3...5 десятичных цифр и довольно редко – с мантиссами в 10...15 цифр. Вспомним, что один байт памяти – это 256 возможных значений, два байта – 65536, т.е. два байта – это примерно четыре с половиной значащие цифры. Явно недостаточно. Если же взять три байта, то они дадут 7 гарантированных десятичных цифр. Получается четыре байта вместе с байтом знаков и порядка, что согласуется с размером ячейки для длинных целых.

Вещественные с длинной мантиссой до 15 значащих цифр обеспечиваются удвоением количества байтов, отводимых под число, причем четыре дополнительных байта отдаются целиком мантиссе. Этим способом достигается так называемая двойная точность вычислений.

В список машинных операций над вещественными включают четыре арифметических действия, команды нахождения абсолютной величины, знака числа, удлинения и укорачивания, преобразования из внутреннего представления во внешнее и обратно. Часть этих команд тоже реализуется "аппаратно", а часть – "программно".

Названные типы элементов информации – текстовые, целые и вещественные – составляют так называемое базовое множество типов.

Обычно в него включают еще один тип – логический, или булевский. Под логическими значениями (своеобразными числами, поскольку их можно вычислять) понимают пару противоположных по смыслу утверждений: "истина" и "ложь". Или их эквиваленты: "да" и "нет", "true" и "false", "YES" и "NO". Размер букв и язык (русский, английский или иной) здесь не играют роли.

Чтобы сохранить логическое значение в памяти, достаточно всего одного бита. В нем можно значение "истина" кодировать единицей, а "ложь" – нулем. Часто именно так и делается. Иногда логические значения кодируют буквами T и F, занимая тем самым уже не бит, а байт.

Может показаться, что логический тип данных в повседневной жизни встречается как будто реже, чем другие. На самом же деле мы просто не обращаем внимания на такие элементарные по своему "устройству" логические значения. А они появляются всегда, когда проводится какое-то количественное сравнение. Например, сортируя группу однородных предметов по длине или весу, мы проводим множество сравнений пар предметов на "больше" и "меньше" и получаем ответы "да" и "нет". Если мы хотим, чтобы ЭВМ могла проводить подобные элементарные логические рассуждения, необходимо, чтобы в составе команд, манипулирующих с целыми, вещественными и символьными операндами, были команды сравнения на строгое равенство (совпадение), строгие и нестрогие неравенства ($<, >, \leqslant, \geqslant, \neq$), вырабатывающие в качестве результата значения "да" в случае истинности отношения и "ложь" – при его нарушении. Языковые конструкции, использующие эти команды (условные операторы, циклы), встречаются практически в любом языке программирования. Например, условный оператор вида

если $a<b$, то (вычислить y по первой формуле),
иначе (вычислить y по второй формуле)

соответствует математической записи

$$y = \begin{cases} a+b, & \text{при } a<b, \\ a/b, & \text{при } a \geqslant b. \end{cases}$$

Такой оператор есть в Алголе, Фортране, Бейсике, Паскале, Си и других языках. Правда, в каждом он оформляется несколько по-своему.

А для того, чтобы построить более сложные сравнения, состоящие из нескольких элементарных, необходимо иметь в распоряжении и операции (команды) с логическими операндами. Известно, что минимальный комплект включает всего три операции, которые называются "и", "или" и "не", или конъюнкция, дизъюнкция и отрицание.

Пример использования операции "и":

если $a<b$ "и" $k=5$, то (вычислить y по первой формуле)
иначе (вычислить y по второй формуле),

что соответствует математической записи:

$$y = \begin{cases} \text{первая формула при } a < b \text{ и одновременно } k = 5, \\ \text{вторая формула при } a \geqslant b \text{ при } a \geqslant b \text{ или } k \neq 5. \end{cases}$$

Еще одна группа команд, обязанная присутствовать в нашем списке, это команды ввода-аывода, выполняя которые машина может копировать (читать) информацию – программы, данные – в свою память и выводить (печатать, рисовать) ее из памяти на дисплей, бумагу и вообще любое внешнее по отношению к центральной части ЭВМ устройство.

Подводя итог рассуждениям о командах, мы неизбежно приходим к важному выводу, что в составе ЭВМ обязательно должно быть устройство, которое умеет выполнять арифметические операции (команды) и операции преобразования над целыми и аещественными, операции сравнения с выработкой лигических значений, по крайней мере, три операции над погическими значениями, команды ввода и вывода информации. Такое устройство называется арифметико-погическим и присутствует в каждой вычислительной машине.

Рассматривая возможные виды информации и операции по ее обработке, мы фактически имеем дело всего лишь с двумя ее категориями – данными и программами. Данные – это тот материал, который обрабатывается машиной. А программа – это перечень команд, описывающий понятным машине образом процесс обработки. Другого рода информации в ЭВМ просто не бывает. Мы не будем здесь вникать в особенности внутреннего устройства блоков, из которых состоит компьютер, и разбираться, какие в них циркулируют сигналы и к какой категории информации эти сигналы следует относить. Речь идет о том, что вводит в машину человек и что он от нее получает. Это – данные и программы.

Любителям компьютерных игр будет, вероятно, интересно узнать, что названные фундаментальные категории любой информации, попадающей в машину, – программы и данные – имеют место и в играх. Когда пользователь, запустив программу, управляет игровой ситуацией, нажимая клавиши-стрелки или отклоняя ручку джойстика, он фактически почти непрерывно вводит все новые и новые данные в программу. Это коды нажатых клавиш, или числа, означающие величину отклонения ручки от вертикали.

Для получения управляющего воздействия в любой случайный момент времени в игровой программе приходится циклически, с частотой нескопько раз в секунду опрашивать состояние буфера клавиатуры, т.е. считывать из него код очередной нажатой клавиши, анализировать и направлять развитие игровой ситуации по одному из нескольких возможных путей в зависимости от кода.

Перечисленные выше базовые типы данных долгие годы составляли близкую к 100% долю от всей информации, хранимой и перерабатываемой на ЭВМ. И только в последнее десятилетие с ростом объемов памяти ЭВМ, появлением множества принципиально новых устройств для ввода и вывода изображений произошло бурное развитие компьютерной графики. Совершенствование аппаратных и программных средств графики превратилось в одно из основных направлений развития ЭВМ. Но поскольку тема графики слишком обширна и богата, в этой книге мы коснемся лишь поверхностно. А заинтересованному читателю посоветуем обратиться к имеющейся литературе по машинной графике.

Существуют два принципиально разных способа построения изображений – отдельными линиями (штриховой или векторный) и сплошным заполнением (растровый). При векторном способе изображение на экране дисплея или листе бумаги строится из отрезков прямых линий (векторов) и, возможно, дуг окружностей различных радиусов. Хранение таких изображений в памяти ЭВМ не вызывает каких-либо проблем, так как, например, для полного определения отрезка прямой на плоскости достаточно указать четыре числа – координаты двух его концов. А на непрерывной ломаной можно сэкономить почти вдвое, поскольку промежуточные точки дстаточно задавать своими координатами однократно. Поэтому для изображения, состоящего из сотни отрезков прямых, потребуется указать не более четырехсот чисел – координат точек концов. Попробуйте нарисовать какую-нибудь картинку, употребив сотню отрезков, и вы увидите, что стопько отрезков – это уже немало. А если взять даже в десятки раз больше отрезков, то объем памяти, необходимой для их координат, будет небольшим по современным меркам.

Читатель, наверное, уже отметил про себя, что наши рассуждения о качестве векторного изображения будут явно неполными, если не принять во внимание толщину линии, ее цвет и яркость. Справедливое замечание. Попробуем сделать оценки. Средняя толщина линии штриха букв в книгах составляет примерно одну десятую миллиметра. Почти такая же и толщина линий штриховых рисунков. На листе бумаги стандартного формата (21x30 см) можно начертить рядом одну к одной самое большее (300 мм/0,1 мм) 3000 линий. Отсюда получаем требования к точности записи координат отрезков. Для стандартного листа можно задавать координаты целыми положительными числами от 0 до 3000, причем изменение на целую единицу будет означать смещение на бумаге на 0,1 мм. Отрицательные числа можно применять, например, для промежуточных точек ломаных. Таким образом, в качестве координат концов отрезков годятся целые числа в двубайтовом представлении, рассмотренные выше, даже если размеры листа будут в 10 раз больше или толщина линии будет в 10 раз меньше. Для экранного изображения с запасом хватит диапазона двубайтовых целых, поскольку большинство современных дисплеев отображает не более 1000x1000 точек.

Здесь надо упомянуть еще об одном существенном моменте, который мы пока только подразумевали. При изображении любого отрезка необходимо проставить точки не только на его концах, но и вычислить координаты всех промежуточных точек, а также проставить (нарисовать) их. Это простая задача линейной интерполяции, которая сводится к многократному решению обыкновенных пропорций, изаестных из школьного

курса математики. Поэтому нет никакой необходимости хранить координаты всех промежуточных точек отрезков, ЭВМ успевает их вычислять прямо по ходу рисования.

Между векторным изображением на экране и векторным изображением на бумаге есть, по крайней мере, три существенных отличия, обусловленных технической природой способов их построения.

Первое отличие заключается в том, что для изображения на бумаге в общем случае требуется больше точек, чем можно"втиснуть" на экран. Но зато уже прорисованные отрезки можно совершенно "забыть". А полностью построенное экранное изображение приходится целиком хранить в специальной буферной памяти (видеопамяти) и воспроизводить на экране с частотой кадровой развертки, т.е. 25–50 раз в секунду в течение некоторого времени, чтобы его можно было успеть рассмотреть.

Второе отличие состоит в том, что на экране технически проще менять яркость линии (точки), а на бумаге – цвет линии (цвет чернил).

Третье отличие – в том, что скорость построения отрезка на экране ограничена быстродействием электроники (вспомним о необходимости вычисления промежуточных координат), а скорость черчения линии на бумаге – скоростью вытекания чернил из пера и инерционностью механических деталей графопостроителя. В силу этих причин экранная графика работает намного быстрее бумажной (об устройствах построения векторных изображений см. раздел "Периферийные устройства").

Растровая графика отличается от векторной тем, что выделенное для изображения пространство – прямоугольник на бумаге или экране – заполняется сплошь, а не отдельными линиями. Это, на первый взгляд, не очень существенное отличие приводит на практике к цвлому ряду изменений в устройствах вывода изображений.

Во-первых, резко возрастает необходимый обьем памяти, так как в случае растровой графики надо хранить какой-то минимум информации по каждой точке прямоугольника изображения, например, уровень яркости или номер цвета. Обычно ограничиваются 8-ю уровнями яркости или 8...16-ю цветами. Соответственно каждая точка требует 3–4 бита памяти. Для изображения размером в полный стандартный лист с диаметром точки 0,1 мм, как нетрудно подсчитать, требуется несколько мегабайт памяти. С растровыми дисплеями ситуация несколько полегче, размеры видеопамяти для них нужны, по существу, те же, что и для векторных дисплеев.

Во-вторых, чтобы нанести на лист несколько миллионов мелких точек, в общем случае каждую со своим цветом, требуется иной физико-механических принцип, чем тот, который применяется в перьевых графопостроителях.

Итак, читатель получил прдставление о типах элементов, составляющих информацию, и отдельных операциях, которые должна уметь выполнять над ними ЭВМ. Для полноты картины рассмотрим в целом процесс решения любой задачи, осуществляемый с помощью компьютера.

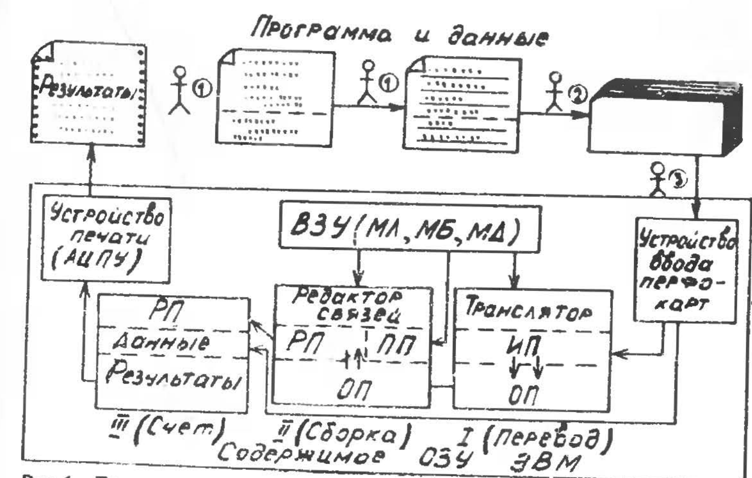


Рис.1. Процесс прохождения задачи с использованием перфокарт: ИП – исходная программа; ОП – объектная программа; РП – рабочая программа; *1* – программист; *2* – оператор подготовки данных; *3* – оператор ЭВМ

На рис.1 показаны основные стадии традиционного технологического процесса прохождения задачи на ЭВМ. В таком виде этот процесс существовал с момента появления первых алгоритмических языков в конце 50-х годов и до начала 80-х. Проходя по технологической цепочке, исходная программа, записанная на языке программирования первоначально на бумаге, претерпевает несколько превращений, прежде чем ее эквивалент – программа в машинных кодах – исполнится арифметико-логическим устройством ЭВМ. Результатом пропуска задачи являлась распечатка (листинг), в которой выдавался исходный текст с примечаниями транслятора, сообщения об обнаруженных ошибках, сообщения редактора связей о результатах сборки рабочей программы и собственно результаты вычислений, для которых все и делалось.

Такая длинная технологическая цепочка часто давала сбои во многих звеньях; продолжительность ожидания программиста на многих этапах была значительной, поэтому на поиск и устранение неизбежных ошибок е программе уходили дни, а нередко и месяцы. Программисты тратили много времени непроизводительно. Например, при отсутствии доступа к перфоратору отверстия в перфокартах прорезали вручную лезвием безопасной бритвы. Многие энтузиасты знали наизусть перфокарточные коды (комбинации отверстий) символов и могли прочитать текст программы по "дыркам" или расшифровать значения вещественных во внутреннем представлении и перевести его в обычное десятичное и обратно.

К счастью, за последние 7–8 лет ситуация радикально изменилась к лучшему: ЭВМ заметно "поумнели" и уже не требуют от человека выполнения огромного количества рутинной работы. Многие лишние этапы и звенья на пути от записи готового алгоритма до получения результата

его исполнения удалось исключить. Прекращен выпуск перфокарт и соответствующего оборудования. Почти исчезла профессия операторов подготоаки данных. На малых и персональных машинах нет операторов ЭВМ. Ввод исходной информации в машину осуществляется пользователем либо с клавиатуры, либо посредством специальных устройств – сканеров, модемов, устройста связи с техническим объектом и др. Машины постепенно превращаются из загадочных гудящих монстров, занимавших просторные залы, в малогабаритные настольные приборы, обращение с которыми уже не требует от человека множества специальных знаний, далеких от решаемых задач.

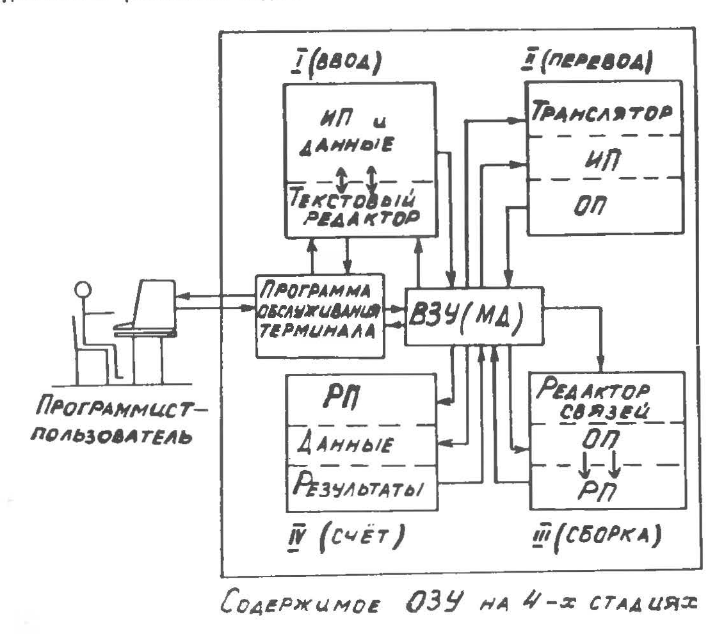


Рис.2. Процесс прохождения задачи с использованием терминала

Модифицированный вариант традиционного на основе языке программирования процесса прохождения задачи выглядит иначе (рис.2). В нем между программистом - пользователем и машиной уже нет лишних звеньев и живых посредников, поэтому тратится во много раз меньше времени на всякие ожидания.

Дальнейшее развитие технологии, ориентированное на ее применение на персональных ЭВМ, пошло по даум направлениям. В одном из них, предназначенном для небольших машин класса "Электроника БК-0010", используется программа-интерпретатор простого языка типа Бейсик. Интерпретатор выполняет также функции текстового редактора и сборщика. В другом направлении произошло объединение трех служебных

программ – текстового редактора, транслятора-переводчика и редактора связей в одну большую систему, так называемую "турбо". Они созданы для многих популярных языков программирования. Турбосистемы занимают, как правило, сотни килобайт оперативной памяти и требуют быстрого процесса, но зато при очень маленьких потерях времени на ожидание они имеют более богатые возможности, чем интерпретатор.

АРХИТЕКТУРА ЭВМ

Под архитектурой ЭВМ в этой книге понимается система команд процессора, структура памяти и набор основных устройств. В самом общем виде структуру любой ЭВМ можно представить в виде схемы (рис.3).

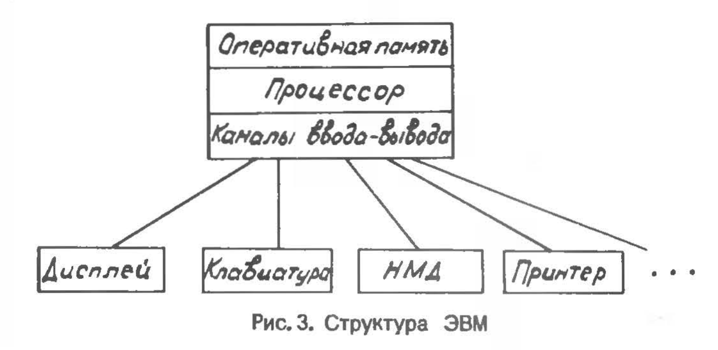


Рис. 3. Структура ЭВМ

Быстрая, или оперативная, память – это важнейший компонент центральной части ЭВМ. От емкости ОЗУ (оперативного запоминающего устройства) зависят все основные характеристики машины – производительность, размеры, энергопотребление, цена. Оперативная память современных компьютеров создается из специальных микросхем памяти и может занимать от нескольких квадратных сантиметров на одной печатной плате до отдельно стоящего большого шкафа.

Слова и байты памяти, состоящие из бит, пронумерованы, начиная с нуля. Этот номер называется адресом. Максимально возможный объем оперативной памяти машины (т.е. ее адресное пространство) зависит от разрядности (длины слова) и принятых технических решений. Чаще всего он составляет 64 или 512 Кбайт, 16 Мбайт. Конкретные экземпляры машин поставляются далеко не всегда с ОЗУ максимальной емкости.

Кроме собственно оперативной памяти в процессоре имеется еще и сверхбыстрая память небольшого обьема. Она представляет собой несколько ячеек, называемых регистрами, которые бывают одинарного (длиной со слово) и увеличенного размероа. Регистры работают а несколько раз быстрее, чем ячейки ОЗУ, и используются для увеличения общего быстродействия машины. На регистрах хранят результаты промежуточных действий и часто используемые значения. В дополнение к регистрам и ОЗУ в состав быстрой памяти включают постоянные за-

поминающие устройства (ПЗУ) объемом в несколько килобайт. В них хранят наиболее часто используемые небольшие программы (например, программу начальной загрузки ядра операционной системы или интерпретатор Бейсика).

Второй компонент центральной части ЭВМ – это процессор, который для потребителя интересен прежде всего своей системой команд и быстродействием, т.е. скоростью их выполнения. Система команд процессора представляет собой набор из нескольких десятков, иногда и сотен отдельных операций, которые умеет выполнять процессор данного типа.

Во всем мире разработано множество различных, несовместимых между собой систем команд, соответствующих отдельным семействам ЭВМ. Чем богаче система команд машины и шире разнообразие выполняемых в ней операций, тем удобнее для такой ЭВМ писать сложную программу и тем меньше ее размер. Но большую систему команд реализует более сложный и, следовательно, более дорогой и менее надежный процессор. Кроме того, сложную систему команд программисту приходится дольше изучать и осваивать. По способу адресации, т.е. использования номеров (адресов) ячеек операндов, команды бывают безадресные, одно-, двух- и трехадресные.

Одной из наиболее простых для воплощения в аппаратуре и ясных для понимания является трехадресная система команд, в которой вся оперативная память делится на одинаковые ячейки, размером 4–6 байт каждая. Нет ни регистров, ни ПЗУ. В одну ячейку записывается только одна команда или одно число (целое или вещественное). Каждая команда состоит из четырех элементов – кода операции и трех адресов. Код операции – это номер операции в таблице команд, например, 001 – сложение вещественных, 002 – вычитание, 003 – умножение, 004 – деление, 005 – безусловный переход на другую ячейку (а не следующую за текущей), 077 – останов процессора – последняя команда программы и т.д. Команды и числа записываются в восьмеричной системе счисления. В этой системе команд программа в машинных кодах для вычисления $X = A + B \times C$ будет выглядеть, например, так:

Номер ячейки	Команда или число	Пояснение	
1	003 0004 0005 0007	$B \times C \to X$	программа
2	001 0007 0006 0007	$X + A \to X$	
3	077 0000 0000 0000	останов	
4	037 2130 0000 0000	B	заданные числовые значения
5	040 5312 7564 0000	C	
6	104 3214 6541 2300	A	
7	000 0000 0000 0000	X	ячейка результата

Первая команда расшифровывается следующим образом: "взять содержимое ячеек, заданных первым адресом (0004 – там находится B) и вторым адресом (0005 – там находится C), перемножить друг на друга и результат записать в ячейку, указанную в третьем адресе (0007 – там будет X, а пока в неё занесем промежуточный результат)".

Вторая команда: "взять числа из ячейки 0007 (там уже появилось произведение $B \times C$) и ячейки 0006 (туда мы записали значение A), сложить вместе и сумму записать в ячейку 0007 (теперь это результат вычисления по формуле)".

Третья команда: "остановить процессор, так как все сделано".

В три последующие ячейки (с номерами 4, 5 и 6) мы должны до запуска программы занести исходные данные – числовые значения величин A, B и C во внутреннем представлении, которое "понятно" процессору. О нем говорилось в предыдущем разделе.

В нашей простейшей программе нет команд ввода-вывода, организации циклов и условных операторов, вызова подпрограмм, тем не менее она уже дает представление о том, как примерно выглядят система команд и программа для трехадресного процессора. Видна и недостаточная эффективность трехадресной системы. Например, результат первой команды можно было бы и не отсылать в память, поскольку он используется в следующей команде, а оставить в сверхоперативной памяти (на регистрах), если бы она была. Команда безусловного перехода, не показанная здесь, имеет всего один адрес – номер ячейки, на которую надо "передать управление". К исполнению команды, находящейся в ней, процессор перейдет, выполнив команду перехода (безусловной передачи управления). Значит, часть памяти – больше половины ячейки – в этой команде не может быть использована.

Во многих случаях более удобна система команд, в которой предусмотрено применение быстрых регистров. В ней задача вычисления $C = A + B$ распадется на группу машинных команд: 1) прочитать число из ячейки A и поместить на первый регистр; 2) прочитать число из ячейки B и поместить его на второй регистр; 3) сложить арифметически содержимое первого и второго регистров, результат оставить на первом регистре; 4) записать содержимое первого регистра в ячейку C. Необходимая длина машинного слова для записи таких команд значительно меньше, чем в случае трёхадресной системы.

Большинству современных пользователей ЭВМ более подробные знания системы команд не требуются, поэтому ограничимся сказанным и перейдем к очень важному для практики понятию "быстродействие". Его указывают в одной из трёх единиц измерения: число операций в секунду над вещественными операндами (скорость "плавающей" арифметики), число операций в секунду над содержимым регистров, тактовая частота процессора (в МГц).

Если еще учесть, что разные команды, например, вещественного сложения и деления, требуют существенно разного времени, то станет

ясно, что понятие "быстродействие" – вещь довольно неопределенная. Дополнительную неопределенность вносят системы программирования. Так, операция $C = A + B$ в режиме интерпретации на Бейсике будет выполняться примерно в десять раз дольше, чем в программе, транслированной с Бейсика (подробнее об этом см. раздел "Перевод программ на машинный язык...").

Существует несколько способов сравнения быстродействия ЭВМ. Один из них заключается в том, что подбирается некоторая представительная "смесь", отрвжающая реальную частоту употребления тех или иных операций, например, в научных расчетах. "Смесь" записывается в виде небольшой программы и прогоняется на разных машинах. Сравнение времен прогона даст картину соотношения скоростей. Разновидностью этого способа является сравнение быстродействия конкретного процессора с каким-либо хорошо известным. Специальная программа, зная скорость базового процессора, определит, ао сколько раз данный процессор быстрее или медленнее базового. Причем программа проводит сравнение отдельно по разным типам операций, что позволит получить более детальную картину.

Следующий пример простой тестовой программы ("бенчмарк") заимствован из [19]. Ее текст был воспроизведен на Алголе-ГДР, Алголе-68, Фортране, Паскале и пропущен на нескольких машинах. Для оценки чистого времени счета для 1000 циклов по счетчику K программа прогонялась с разными пределами для K. Полученные результаты сведены а табл.1.

Таблица 1

Время "чистого" счета до $K = 1000$

ЭВМ и язык программирования	Время, с
БЭСМ-6, Фортран-ГДР	0,080
БЭСМ-6, Алгол-ГДР	0,086
БЭСМ-6, Фортран-ДУБНА	0,23
БЭСМ-6, Паскаль	0,24
ЕС 1055М, СВМ, Фортран ОЕ	0,11
ЕС 1055М, СВМ, Фортран VS	0,11
ЕС 1055М, СВМ, Фортран-77	0,22
ЕС 1055М, СВМ, Алгол-68 (стандартные режимы)	0,38
ЕС 1055М, СВМ, Алгол-68 (отключен контроль)	0,17
ЕС 1055М, СВМ, Паскаль	0,42
СМ 1420, Фортран	0,45
ДВК (процессор МС 1201.02). Фортран	0,9
ДВК (процессор МС 1201.02). Паскаль	4,8
ЕС 1840, Турбо-Паскаль	2,5
IBM PC/AT, 16 МГц, Сопроцессор, Турбо-Паскаль	0,37

Ниже приведен текст программы на Паскале и Фортране. Можно самостоятельно проверить и, быть может, уточнить результаты автора.

```
PROGRAM C (OUTPUT);
LABEL 1,2,3,5;
VAR L,K:INTEGER;  A:REAL;
    M:ARRAY [1..5] OF REAL;
BEGIN WRITELN('начало '); K:=0;
1:  K:=K+1;  A:=K/2*3+4-5;  GOTO 2;
3:  FOR L:=1 TO 5 DO M[L]:=A;
    IF K<1000 THEN GOTO 1;
    GOTO 5;
2:  GOTO 3;
5:  WRITELN('конец ');
END.
```

Необходимо отметить, что переход на метку 2 и возврат от нее вверх по тексту к метке 3 есть имитация вызова подпрограммы на Бейсике. Вместе с тем обеспечено участие оператора перехода в затратах времени. Таков смысл использования GOTO 2 и GOTO 3, на первый взгляд, совершенно ненужных. Текст на Фортране в СВМ ЕС:

```
      REAL *4 M(5)
      PRINT 11,K
11    FORMAT (' начало ',I8)
      K=0
 1    K=K+1
      A=K/2*3+4-5
      GOTO 3
 2    DO 4 L=1,5
 4    M(L)=A
      IF (K.LT.1000) GOTO 1
      GOTO 5
 3    GOTO 2
 5    PRINT 12,K
12    FORMAT (' конец ',I8)
      STOP
      END
```

Полученные результаты дают представление о соотношении быстродействия разных машин и эффективности систем программирования в плане затрат процессорного времени на вычислительных задачах. Следует отметить, что данная программа из-за своей простоты не учитывает ряд важных моментов. Во-первых, в ней отсутствует вызов под-

программ (процедур) – операция, встречающаяся практически в любых программах на любых языках. Во-вторых, представленная в программе "смесь" лишь весьма приближенно отражает реальное соотношение частоты выполнения отдельных операций. За счет этой неточности могут возникать заметные расхождения в оценках времени исполнения программ на разных машинах. В-третьих, указанные в табл.1 цифры не позволяют что-либо сказать о времени работы внешних устройств. Их влияние полностью и намеренно исключено, поскольку автору хотелось сравнить только быстродействие разных процессоров. Попутно удалось оценить зависимость его от языков и систем программирования.

А теперь представим себе, что у нас есть на выбор две ЭВМ, отличающиеся одна от другой только тем, что первая имеет в два раза большую память, чем вторая, у которой зато процессор и память работают вдвое быстрее. Цены машин одинаковы, операционная система одна и та же. Какую же ЭВМ предпочесть? Попытаемся разобраться.

Вариант 1. Задачи на нашей ЭВМ будут решаться самые разные, готовых программ для них еще нет, предстоит их написать. В этом случае годится любая машина, так как большинство алгоритмов можно оптимизировать либо по занимаемой памяти за счет некоторой потери быстродействия, либо по скорости за счет памяти.

Вариант 2. Решаемые задачи из одного класса характеризуются сложными повторяющимися вычислениями. Каждый прогон программы требует много времени (десятки минут, часы). Объемы программ и используемых данных сравнительно невелики. Для этого случая лучше взять ЭВМ с более быстрым процессором.

Вариант 3. Решаемые задачи не отличаются сложной математикой, но велик объем обрабатываемой информации (например, сложное редактирование больших текстов). Здесь скорость процессора скажется незначительно, а вот большая память пригодится.

Быстродействие процессора и объем оперативной памяти, вообще говоря, не зависят друг от друга. Но реально для каждого типа процессора есть область оптимального объема памяти. Упрощенно можно считать, что скорость процессора (количество операций в секунду) должна быть численно равна размеру памяти, выраженному в байтах. Большие отклонения от этого правила нежелательны. Если, например, быстродействие процессора составляет 500 тыс.операций/с, то оперативная память должна иметь объем примерно 500 Кбайт. Если объем памяти будет значительно больше этого, то она будет использоваться неэффективно (напомним, что быстрая память – вещь довольно дорогая). При недостатке памяти плохо используется мощность процессора. Кроме того, от программиста при этом требуется много усилий по экономии оперативной и активному применению внешней памяти. Для средних и больших машин, основным режимом работы которых является многозадачный (одновременное выполнение нескольких программ), размер памяти увеличивают по сравнению с указанным в несколько раз.

В персональных ЭВМ при наличии большой памяти часть ее, если позволяют используемые программы, применяют для временного размещения виртуального магнитного диска. Это дает возможность уменьшить время реакции машины и заметно ускорить работу пользователя.

На быстродействии ЭВМ сказывается и скорость работы оперативной пвмяти. Если на засылку данных или команд и чтение их из памяти уходит много времени, то нет смысла добиваться от процессора большого быстродействия. Если на собственно сложение двух чисел уходит одна микросекунда, а чтение слагаемых из памяти и запись суммы обратно занимают 100 мкс, то налицо явное несоответствие в скоростях работы главных устройств машины. На практике быстродействие процессора и памяти тщательно согласовывают.

В нашей стране выпускается несколько основных семейств ЭВМ: от сложных и дорогих до простых и дешевых. Это "Эльбрус-1" и "Эльбрус-2", ЕС ЭВМ ("Ряд-1", "Ряд-2" и "Ряд-3"), многочисленное семейство СМ – ДВК – "Электроника-60" с разновидностями, различные персональные ЭВМ от школьных и бытовых до профессиональных. Машины других семейств и архитектур менее известны и здесь не рассматриваются.

К одному семейству относятся машины, программно совместимые между собой, т.е. программы в машинных кодах, разработанные для какой-нибудь ЭВМ семейства, будут работать и на любой другой машине данного семейства. Программная совместимость компьютеров – очень важное свойство, и о нем следует помнить при заимствовании и передаче программ.

Важнейшие параметры некоторых машин даны ниже, а для получения подробных характеристик следует обращаться к справочникам (см. список литературы).

Основныв характеристики некоторых моделвй ЕС ЭВМ третьвй очервди ("Ряд-3")

	ЕС 1036	ЕС 1046	ЕС 1066
Производительность для научно-технических задач, млн операций/с	0,4	1,3	Более 5
Емкость оперативной памяти, Мбайт	2...4	4...8	8...16

На персональные машины массового применения, о которых в основном и будет идти речь, установлен ГОСТ 27201-87, в соответствии с которым все персональные ЭВМ делятся на пять классов ПМ1, ПМ2,...,ПМ5 в зависимости от 1) разрядности основного процессора (от 8 до 32 бит); 2) быстродействия (от 0,5 до 4 млн. операций/с типа "регистр-регистр"); 3) емкости оперативной памяти (от 0,064 до 8 Мбайт); 4) типа и объема внешней памяти (на гибких дисках – от 360 до 3000 Кбайт), на жестких дисках – от 10 до 80 Мбайт ; 5) числа адресуемых точек для отображения на экране (от 640х200 до 720х512); 6) цветности изображения(одноцветное или монохромное многоцеетное); 7) потребляемой мощности (от 20 до 200 Вт); 8) массы (от 3 до 20 кг).

Наиболее простые ПМ1 предназначены для индивидуального применения в быту, ПМ2 – в качестве рабочих мест учащихся, ПМ3 – для профессиональной деятельности и обучения, ПМ4 – для сложной профессиональной деятельности (научной, инженерной, административно-управленческой, финансовой), ПМ5 – для автоматизации проектирования, научных исследований и других наиболее сложных задач. Цена ПЭВМ составляет от сотен рублей для ПМ1 до десятков тысяч для ПМ5 в минимальном комплекте.

Многие модели ПЭВМ выпускаются в нескольких модификациях, отличающихся наличием цветного монитора, объемом оперативной и внешней памяти, тактовой частотой, дополнительными блоками и устройствами, вариантами операционной системы.

Основные характеристики некоторых персональных ЭВМ приведены в табл. 2.

Таблица 2

Основные характеристики некоторых персональных ЭВМ

Наименование характеристики	ЕС 1840	ЕС 1841	"Нейрон И9.66"
Быстродействие, млн операций/с типа "регистр-регистр"	1,0	1,0	1,0
Оперативная память, Кбайт	256...640	512...1024	256...1024
Память на гибких дисках, по 2 накопителя, Кбайт в каждом	360...720	360...720	360
Память на жестких дисках, Мбайт	–	10	10
Монохромный или цветной монитор с разрешающей способностью, точек	640x200	640x200	640x200

Продолжение табл. 2

Наименование характеристики	"Искра-1030"	ДВК-4	"Электроника-85"	СМ 1910
Быстродействие, млн операций/с типа "регистр-регистр"	1,0	1,0	0,6	1,0
Оперативная память, Кбайт	256...1024	1000	512	512
Память на гибких дисках, по 2 накопителя, Кбайт в каждом	360	512	360	1000
Память на жестких дисках, Мбайт	–	10	5	30
Монохромный или цветной монитор с разрешающей способностью, точек	640x200	800x240	800x240	640x480

Модели ПЭВМ, совместимые с IBM PC, допускают включение в их состав математического сопроцессора, выполняющего арифметические операции над числами с "плавающей" точкой (вещественными – в терминологии языков программирования), что позволяет в несколько раз повысить быстродействие машины на задачах вычислительного характера.

Число строк и символов на экране во всех моделях одинаково и составляет 25x80. Конфигурация машин может несколько отличаться от указанной. Например, автору довелось иметь дело с ЕС 1841 выпуска конца 1989 года, так называемой 7-й комплектации: с черно-белым дисплеем и без жесткого диска, но с максимальной оперативной памятью и манипулятором "мышь", который в технической документации почему-то называется "колобок". Подобные "сюрпризы", к сожалению, не редкость, поэтому перед приобретением машины рекомендуем заранее еще раз выяснить, что входит в ее комплект. Заметим также, что в указанных ПЭВМ из оперативной памяти в любом случае пользователю на запуск программы отводится не более 600 Кбайт.

Наиболее распространенной ОС на машинах, аналогичных IBM PC (ЕС 1840, ЕС 1841, "Нейрон", "Искра-1030, 1031", "Правец-16", "Правец-24", СМ 1910), является MS DOS и ее аналоги. Популярность машин семейства IBM PC в нашей стране очень быстро растет.

ПЕРИФЕРИЙНЫЕ УСТРОЙСТВА

В этом разделе описаны периферийные, т.е. внешние по отношению к центральному блоку – процессору и оперативной памяти – устройства, которыми комплектуются ЭВМ. От набора периферийных устройств, их технических характеристик во многом зависит конкретная отдача от компьютера, сфера его применения.

Клавиатура

Клавиатура, входящая в комплект персональной ЭВМ и терминала больших машин, является на сегодня основным средством ввода информации в машину человеком. И положение вряд ли существенно изменится в ближайшие годы. Любому, кто лично работает с компьютером, приходится теперь иметь дело с клавиатурой. Познакомимся с этим устройством подробнее.

Блок клавиатуры имеет вид плоской коробки из металла или пластмассы, в верхнюю поверхность которой встроены 60...100 клавиш и несколько лампочек-индикаторов. С нижней стороны коробки могут быть вмонтированы откидные упоры, позволяющие при желании фиксировать клавиатуру в наклонном положении. Внутри коробки содержатся электронные компоненты, обеспечивающие формирование кодов клавиш и связь с процессором. Механизм замыкания контактов в клавишах бывает разных типов, от его надежности очень существенно зависит удобство в работе. В дешевых машинах иногда применяются жесткие сенсорные клавиши,

которые не имеют хода и срабатывают от прикосновения. Такие клавиатуры довольно неудобны, но при отсутствии других можно пользоваться и ими. Кнопочные клавиши обычного исполнения очень ненадежны, их также нельзя рекомендовать для профессионального применения.

Наиболее широко используются емкостные и герметизированные контакты. Их устройство довольно сложное, поэтому цена клавиатурного блока при большом числе клавиш составляет заметную долю в полной стоимости персональной ЭВМ.

Глубина хода клавиш менее существенна, чем надежность контакта, но также имеет значение при интенсивной работе. Большой ход позволяет более свободно и быстро "стучать". При малом ходе уже незначительное прикосновение к клавише вызывает ее срабатывание, что нередко ведет к ошибкам. Поэтому наиболее популярны высокие ("скульптурные", или "профильные") клавиши, имеющие большой ход и позволяющие легко чувствовать ошибочное нажатие одним пальцем одновременно нескольких. Для профессионалов имеют значение даже такие "пустяки", как жесткость пружины, возвращающей клавишу, или бесшумность нажатия. Полезны для скоростного набора текстов выпуклые точки или черточки, которыми иногда снабжаются отдельные символьные клавиши. Они помогают ориентироваться при работе "вслепую".

Некоторые стандарты предусматривают разные высоту и наклон поверхности клавиш, чтобы в совокупности эти площадки образовывали цилиндрически вогнутую поверхность. Клавиши верхнего и нижнего рядов должны быть более высокими, их верхние площадки должны иметь наклон к среднему, менее высокому ряду. Клавиши промежуточных рядов имеют промежуточную высоту. Подобное строение имеет большинство клавиатур зарубежного производства, в том числе широко известного в нашей стране японского школьного комплекта учебной вычислительной техники "Ямаха".

Взаимное расположение большей части символьных клавиш регулируется несколькими стандартами, принятыми для пишущих машин и отражающими частоту употребления букв алфавита в разных языках – русском, английском, немецком и др. Главная идея таких стандартов одна и та же: чем чаще буква встречается в языке, тем ближе к центру клавиатуры ее помещают. Коротко стандарт обозначают перечислением леаого края верхнего ряда букв. Так, стандарт для русского языка называют ЙЦУКЕН. В соответствии с ним выполняются клавиатуры для массовых ЭВМ, выпускаемые в нашей стране и для нас за рубежом. Для языков на основе латинского алфавита наиболее распространен стандарт QWERTY. Реже встречаются другие варианты расположения, отличающиеся от QWERTY размещением нескольких крайних букв.

На клавиатурах с кириллицей, размещенной по ЙЦУКЕН, наносят, как правило, и латиницу. Здесь возможны два варианта – традиционный и современный. Традиционный вариант расположения латинских букв следует принципу их зрительно-звукового подобия русским – JCUKEN, т.е. на каждой клавише к русской букве добавляется аналогичная ей ла-

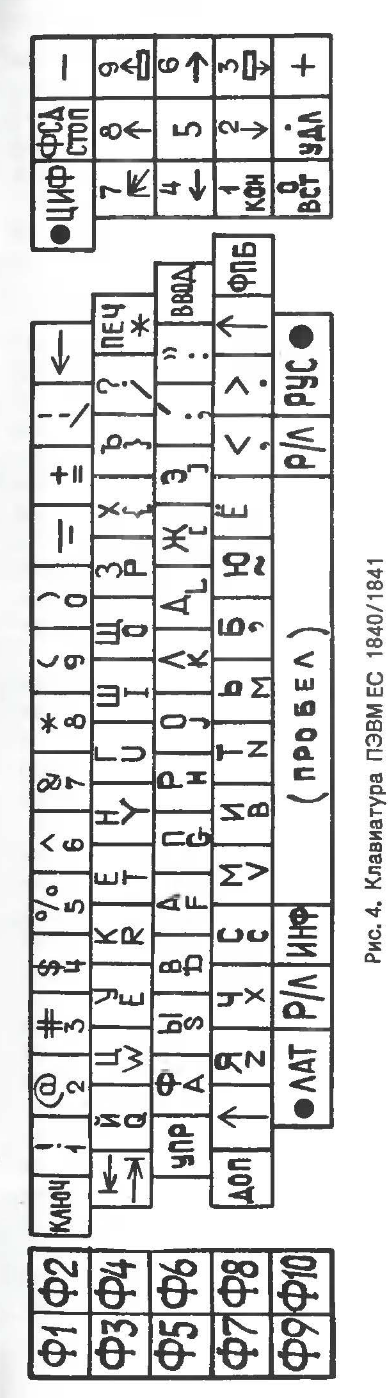


Рис. 4. Клавиатура ПЭВМ ЕС 1840/1841

Соответствие русских и английских обозначений управляющих клавиш

Ф1...Ф10 – F1...F10	← – Backspace	⇱ – Home
Ключ – Esc	Печ – PrtScr	Кон – End
⇆ – Tab	Ввод – Enter ↵	⇧ – PgUp
Упр – Ctrl	↑ – Shift	⇩ – PgDn
Доп – Alt	Циф – Num Lock	Вст – Ins
ФПБ – Caps Lock	ФСД СТОП – Pause Break	Удл – Del
Пробел – Space	Серый минус – Grey–	Серый плюс – Grey+

тинская. Так наши произаодители клааиатур размещали латиницу вплоть до появления персональных ЭВМ. Вариант JCUKEN был удобен и устраивал большинство пользоаателей, а их было гораздо меньше, чем сейчас.

В клааиатуре персональной ЭВМ ЕС 1840 (рис.4) аперавые применен принцип строгого аыполнения двух общепринятых языкоаых стандартоа – ЙЦУКЕН и QWERTY. Этого принципа придерживаются иностранные произаодители и соаместные предприятия, предлагающие ПЭВМ для потребителей в СССР. На клааиатурах "персоналок", на которых изначально отсутстаует кириллица, ее надписывают по ЙЦУКЕН, поскольку сущестаующие русскоязычные драйверы поддерживают именно это расположение.

Цифры обычно располагают над букаами а последоаательности 1,2,...,9,0. На цифровых клааишах, а также слева и справа от них размещают знаки препинания, знаки арифметических действаий, круглые, квадратные и фигурные скобки и некоторые другие симаолы. В расположении их наблюдается большой разнобой.

Управляющие клавиши помещают аокруг символьных, аыделяют цаетом, их состаа и местоположение, если иметь в аиду клавиатуры ЕС 7927 (для ЕС ЭВМ), СМ 7209, ВТА 2000, ДВК 1,2,3, Видеотон-340, VDT 52100, IBM PC/XT/AT, IBM PS/2, ЕС 1840/1841, очень сильно отличаются.

Можно назвать асего одну клааишу, которая на клааиатурах всех ЭВМ аыглядит и располагается одинакоао и не подписыаается – клааиша пробела ("space") – самая длинная в нижнем ряду.

Символьные клааиши чаще всего саетлые. Черный цает с белыми надписями менее удобен, так как при аерхнем осаещении дает сильные блики.

Все клавиатуры имеют стрелки смещения курсора, по крайней мере, в направлениях ааерх-вниз, влеао-апраао. Но их расположение очень произвольно. Быаают и дополнительные клааиши управления курсором, например, стрелка переаода а начало строки или в леаый аерхний угол экрана ("домой" – "home"). Эти клааиши аыделяют цаетом или выносят в отдельную группу. При длительном нажатии клааиши перемещения курсор бежит по экрану со скоростью, которую можно регулироаать.

Упрааляющие клааиши, в составе которых обязательно есть клааиша исполнения ("ааод" или "ВК" – аозарат каретки) и фиксации верхнего регистра, аыделяются цветом и располагаются спрааа и слева от симаольных клааиш. Дополнительные упрааляющие клавиши, как а ДВК-1, ДВК-2, могут быть расположены группами а аерхней части клавиатуры.

Программируемые функциональные клавиши – это особый тип клавиш, функции которых можно менять анутри исполняемой программы. В этот набор аходят 10...20 клавиш, аыделенных цаетом и расположенных группой слеаа, спрааа или аерхним отдельным рядом. Клавиши этого типа очень удобны как дополнительные управляющие, смысл которых можно устанавлиаать по саоему усмотрению. Многие зарубежные диалогоаые программы, особенно для ПЭВМ, используют функциональные клавиши.

Практически все клавиатурные блоки имеют саетоаую и (или) заукоаую сигнализацию. Световая обеспечиаается лампочками, образующими отдельный ряд, как а ДВК-1,2, или астроенными прямо в клавиши (ЕС 1840/1841, "Ямаха"). Обычно они сигнализируют о включенном регистре и алфааите, необходимость а других зависит от конструкции клааиатуры и способа связи ее с процессором и дисплеем.

Звукоаая сигнализация выполняется а аиде миниатюрного динамика (как в карманном радиоприемнике). Заук разной тональности может саидетельстаовать о замыкании контакта а клааише или о блокировке клааиатуры. Его громкость можно регулировать либо отключать соасем. В основном блоке ПЭВМ и дисплее терминала ЕС 7927 имеется второй динамик, его назначение – выдача сигналов при работе ЭВМ, а не пользователя.

Существует интересный аариант замены (имитации) клааиатуры а персональных ЭВМ манипулятором "мышь" и специальной программой, рисующей на экране отсутствующую клавиатуру. Когда надо нажать на клавишу, пользоаатель устанавлиаает "мышью" курсор на её изображение и нажимает кнопку манипулятора. Программа "исполняет" нажатие соответстаующей клавиши. Даигая курсор по изображению на экране и нажимая только одну кнопку, можно в принципе аообще обходиться без реальной клааиатуры. Правда, скорость ввода текста при этом аряд ли приблизится к скорости обычного ааода.

Такой способ можно рекомендоаать для бытоаого применения в том случае, когда ценой экономии денег на клааиатуре можно терпеть некоторые неудобства а работе с ПЭВМ. Мы полагаем при этом, что манипулятор "мышь" и копия программы-имитатора стоят заметно меньше реальной клааиатуры.

Печатающие устройства

Основным способом передачи результатов работы от ЭВМ челоаеку на протяжении всей истории аычислительной техники был аыаод их на бумагу. Промышленностью разных стран выпускается множестао типоа печатающих устройств (принтероа), осноаанных на нескольких базоаых физических принципах. Используемые а принтерах, которые раньше назыаали АЦПУ – алфааитно-цифровое печатающее устройстао, электромеханические способы нанесения краски на бумагу определяют как разнообразие и качестао печати, так и массу, габаритные размеры, произаодительность и цену принтероа.

Наиболее распространенныв типы печатающих устройста можно разделить по принципу действия на следующие группы: а) барабанные; б) со сменным шрифтоносителем – лепесткоаые ("ромашка") и "бочоночные"; а) мозаичные (матричные, точечные, игольчатые); г) лазерные; д) струйные (чернильные).

Барабанные АЦПУ предстааляют собой доаольно большие и тяжелые ящики-тумбы, позволяющие аыдавать любую симаольную информацию на непрерыаную бумажную ленту, обычно шириной 420 мм (две страницы

стандартного формата). Лента по краям имеет отверстия (перфорацию) для протяжки. Устройство и принцип дествия АЦПУ состоят в следующем. Выпуклые зеркально перевернутые символы (буквы, цифры и т.п.) нанесены на торец стального диска диаметром около 10 см, 120...160 одинаковых дисков соединены в барабан, между барабаном и бумагой помещена широкая черная красящая лента, а за бумагой находится линейка из электромагнитных молоточков (рис.5).

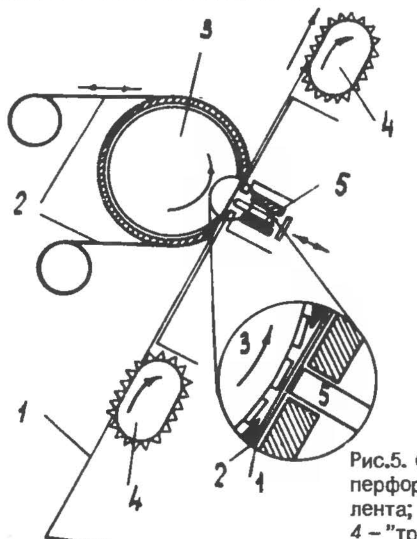


Рис.5. Схема работы АЦПУ: *1* – бумага, перфорированная по краям; *2* – красящая лента; *3* – барабан-шрифтоноситель; *4* – "тракторы" подачи бумаги; *5* – электромагнитный молоточек

При работе АЦПУ барабан непрерывно вращается. Когда напротив молоточка оказывается нужный символ, молоточек ударяет по бумаге, она прижимается к красящей ленте и через неё – к барабану. Получается оттиск символа.

Все молоточки на линейке действуют параллельно, а если печатаемая строка состоит из одинаковых символов, то и строго одновременно. Когда нужные символы на строке отпечатаны (а для этого требуется в худшем случае один полный оборот барабана), производится перемещение бумаги на следующую строку. Поскольку диаметр барабана невелик и сделать его больше нельзя, то по его окружности удается разместить не более сотни символов нормального размера. Поэтому в комплект включают лишь заглавные (прописные) буквы русского и латинского (речь об отечественных АЦПУ) алфавитов, арабские цифры, скобки, знаки препинания, знаки арифметических действий и некоторые другие символы. Полиграфическое качество такой печати невысокое. Достоинством являются относительная простота конструкции и большая производительность. АЦПУ до сих пор применяются на больших ЭВМ в вычислительных центрах.

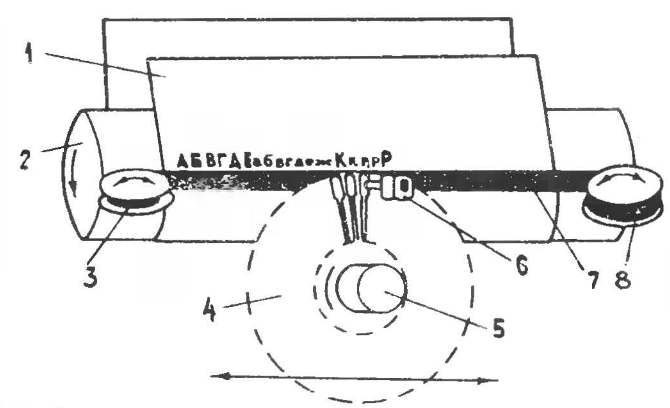


Рис.6. Схема принтера "ромашка": *1* – бумага; *2* – обрезиненный вал; *3,8* – катушка; *4* – сменный диск-шрифтоноситель ("ромашка"); *5* – электродвигатель; *6* – электромагнитный молоточек; *7* – красящая лента

В лепестковых принтерах (рис.6) шрифт нанесен на плоскость, а не на торец диска, разрезанного на лепестки (отсюда и название "ромашка"). Диск изготовлен из упругого материала, поэтому лепестки могут слегка отгибаться. Между диском и бумагой, обернутой вокруг валика, помещается узкая красящая лента. Диск закреплен на оси шагового электродвигателя. Сзади, за верхним лепестком, расположен электромагнит с молоточком. Диск с двигателем и молоточком перемещается вдоль валика, т.е. вдоль строки на бумаге. Когда нужно отпечатать символ, диск рывком поворачивается так, чтобы лепесток, содержащий нужный символ, стал верхним, затем молоточек ударяет по лепестку, который прижимает красящую ленту к бумаге и делает оттиск. Строка формируется строго последовательно, буква за буквой, поэтому производительность такого принтера в принципе не может быть высокой. Зато можно добиться хорошего качества печати при небольших габаритных размерах и цене устройства. Бумага здесь годится любая – рулонная, перфорированная (или нет), листовая (не во всех). Меняя диск-ромашку, можно разнообразить не только шрифт (его размер, гарнитуру, наклон, жирность), но и состав символов, а используя специальную красящую ленту, получать распечатку очень высокого качества. Этот принцип лежит в основе работы некоторых пишущих машинок. Например, оригинал данной книги для типографии отпечатан на такой машинке со сменными дисками-ромашками.

Разновидность таких принтеров – "бочоночные" – имеет сменный шрифтоноситель, выполненный в форме бочонка, на бока которого нанесены выпуклые символы. Перед печатью бочонок поворачивается вдоль своей оси и наклоняется в нужную сторону, чтобы приблизить к месту в строке требуемый символ, а оттиск получается касанием бочонка красящей ленты.

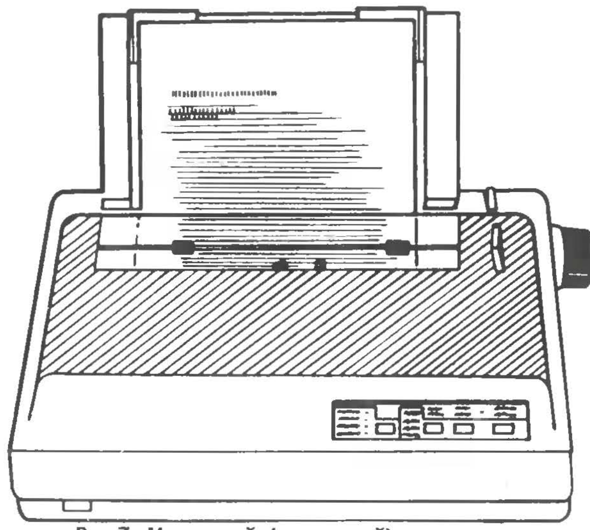


Рис.7. Мозаичный (матричный) принтер

Мозаичные печатающие устройства (рис.7) так же, как и ромашковые, имеют небольшие габаритные размеры, массу и цену и не отличаются высокой производительностью. Качество печати – среднее. В мозаичных принтерах изображение формируется из множества точек, оставляемых ударами тупых иголок по красящей ленте. Благодаря этому они могут выдавать на бумагу не только алфавитно-цифровую информацию, но и рисунки, графики, диаграммы и даже полутоновые изображения. Обычно иголки составлены в вертикальный ряд (в высоту печатаемой строки) в количестве 7...9 штук. Они прижимаются к красящей ленте также электромагнитами (каждая своим), а отводятся назад пружинками. Печатающий механизм с иголками перемещается равномерно вдоль строки бумаги, обернутой вокруг стального или резинового валика. Красящая лента такая же узкая, как у пишущих машин, чаще всего помещена в пластмассовую кассету. Бумага используется листовая, рулонная, с отеерстиями или без таковых. Повторная печать символа с небольшим смещением иголок лучшает качество изображения, точки становятся почти незаметными.

В некоторых мозаичных принтерах применяется широкая многоцветная красящая лента (3–4 цвета), что позволяет ранообразить цветовое оформление распечаток. Размеры мозаичных принтеров непрерывно уменьшаются, а число иголок растет. В последних образцах устройств количество иголок доведено до 24 и даже до 48, а их толщина уменьшена до долей миллиметра, что позволило достичь качестеа печати, сравнимого с качеством, даваемым ромашковыми принтерами. В графическом режиме мозаичные принтеры могут сплошь заполнять пространство на бумаге,

формируя тем самым изображение любого содержания. Благодаря перечисленным достоинствам, мозаичные принтеры находят очень широкое применение, особенно в составе персональных машин. Выпускается множество вариантов и разновидностей мозаичных принтеров, которые называют также матричными (dot matrix printer), игольчатыми и точечными.

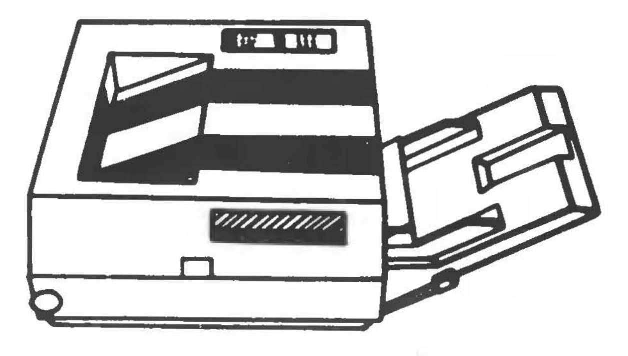


Рис. 8. Лазерный принтер (внешний вид)

Лазерные принтеры (рис.8) – наиболее сложные и дорогие из малогабаритных печатающих устройств. Принцип их действия основан на известном свойстве прилипания измельченной полимерной краски к статически заряженной полупроводниковой поверхности. В лазерном принтере (на рис.9 показан один из возможных вариантое его внутреннего устройства) поверхность цилиндра из полупроводникового материала равномерно по площади заряжается от высоковольтного источника. Затем меняющимся по интенсивности тонким лазерным лучом в нужных местах поверхность разряжается. С помощью специального валика – электромагнитной щетки – пылевидная краска наносится на цилиндр. В тех местах, где заряд остался (луч лазера его не коснулся), пылинки прилипают и далее вращением цилиндра переносятся на бумагу. Другим электрическим полем, действующим с обратной стороны бумаги, частицы краски перетягиваются на нее. Далее под воздействием мощной лампы (на рис. не показана) краска плавится и впитывается в бумагу. Оставшиеся на цилиндре заряды и краска снимаются разряжающими лампами и скребком.

Луч лазера, формирующий изображение, бегает вдоль цилиндра, отражаясь от многогранного вращающегося зеркала. Цилиндр и зеркало вращаются равномерно, а яркость луча меняется под управлением процессора. Точнее, вспышки луча повторяют распределение бит в специально выделенной памяти, в которой процессором с помощью программ печати нулями и единичками формируется будущее изображение. Размер этой памяти должен быть достаточным для построения полной страницы со всеми деталями. Дело в том, что процесс печати в лазерных принтерах имеет особенность: начатую страницу необходимо допечатать до конца без

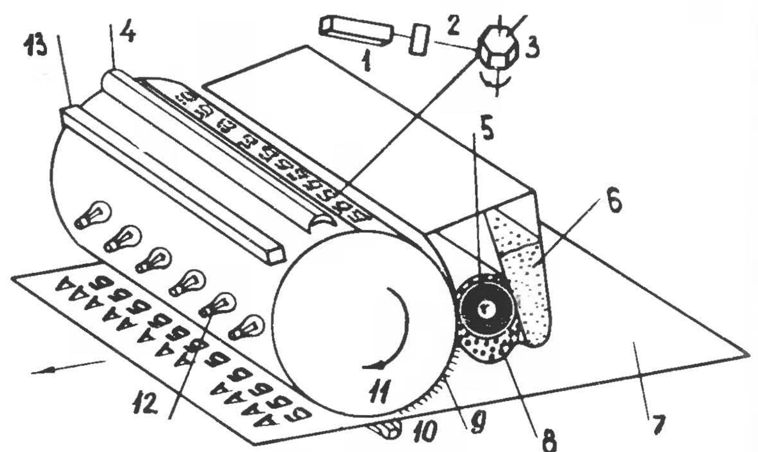
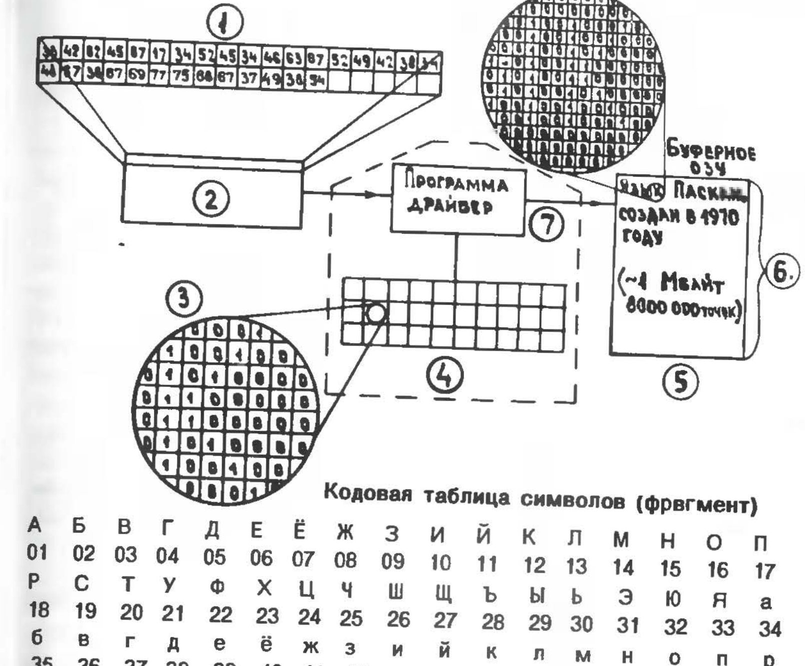


Рис.9. Внутреннее устройство и схема работы лазерного принтера: *1* – лазер; *2* – модулированный лазерный луч; *3* – многогранное вращающееся зеркало; *4* – невидимое изображение; *5* – вращающийся магнит; *6* – резервуар с пылеаидной краской (тонером); *7* – бумага; *8* – железные опилки; *9* – изображение а виде "бохромы" из частиц краски; *10* – приемный коронный разряд; *11* – цилиндр с полупроводниковой поверхностью; *12* – рад разряжающих ламп; *13* – съемный нож; *14* – заряжающий коронный разряд

остановок, в противном случае на ней неизбежно появятся крупные дефекты. Если мы хотим получать изображение высокого качества, т.е. с проработкой мельчайших деталей, то луч лазера должен быть очень тонко сфокусирован, краска должна быть мелко помолота, а буферная память под формируемое изображение – иметь достаточный объем. Его мы уже оценивали, когда вели речь о растровой графике. Он состаалял как минимум сотни килобайт.

Процесс подготовки и вывода информации растровыми периферийными устройствами, к которым относится и лазерный принтер, поясняет рис.10. На нем показана схема формирования точечного изображения страницы текста, который начинается словами "Язык Паскаль создан в 1970 году". Буквы закодированы по таблице, представленной в нижней части рисунка. Строит изображение знакогенератор – программа, связанная с файлом шрифта. Шрифтом яеляется набор матриц бит, в которых единицами на фоне нулей зафиксированы геометрические образы символов. Размеры матрицы – от 5x7 точек-бит для простейшего рубленого шрифта до 30x40 и более для аысококачественной гарнитуры. Для кириллицы требуется около 85 матриц на один типоразмер шрифта. Полагаем также, что страница текста содержит в среднем 3000 симаолов, считая пробелы между словами. Приведенные на схеме цифры достаточно условны. В каждом конкретном случае они могут отличаться в ту или иную сторону. Эта схема работает также при выводе изображения на графические дисплеи, в несколько измененном виде – в мозаичных принтерах с загружаемыми шрифтами.



Кодовая таблица символов (фрвгмент)

А	Б	В	Г	Д	Е	Ё	Ж	З	И	Й	К	Л	М	Н	О	П
01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17
Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	а
18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34
б	в	г	д	е	ё	ж	з	и	й	к	л	м	н	о	п	р
35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51
с	т	у	ф	х	ц	ч	ш	щ	ъ	ы	ь	э	ю	я	␣	0
52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68
1	2	3	4	5	6	7	8	9	...							
69	70	71	72	73	74	75	76	77								

Рис.10. Схема формирования растрового изображения: *1* – уаеличенное начало содержания страницы; *2* – страница из текстового файла; *3* – матрица букаы "К" строчной; *4* – файл шрифта; *5* – 3000 байт смысловой информации; *6* – строчки бит, соответствующих точкам изображения страницы; *4* и *7* – знакогенератор

Для печати на лазерном принтере текстовый файл передается программе-драйверу, который выполняет свою задачу в два приема: 1) готовит изображение в буферном ОЗУ; 2) переводит изображение из буферного ОЗУ на бумагу. В общем виде алгоритм формирования изображения в буферном ОЗУ (БОЗУ) представлен на рис. 10.

1. Очистка БОЗУ путем записи нулей во все биты.

2. Прием кода символа (буквы) из текстового файла. Если файл исчерпан, то переход на п. 7.

3. Поиск соответствующей матрицы изображения символа в файле шрифта.

4. Вычисление места размещения изображения в БОЗУ.

2*

5. Копирование бит матрицы в буферное ОЗУ.

6. Переход на п. 2.

7. Конец подготовки.

Непосредственно печать производится следующим образом (рис.9). Сначала приводятся в действие оптико-механические элементы принтера – лазер, многогранное вращающееся зеркало, полупроводниковый цилиндр, источники высокого напряжения. Затем в темпе движения луча по образующей цилиндра происходит считывание бит вдоль строчек БОЗУ. Биты-нули при этом на луч не влияют, а биты-единицы гасят (перекрывают) луч с помощью специального устройства – модулятора. Следовательно, точки на цилиндре, соответствующие битам-единицам, остаются незасвеченными, и к ним прилипают частицы краски, переходящие затем на бумагу.

Качество полиграфического исполнения с помощью лазерного принтера такое же высокое, как и у лучших ромашковых, а возможности формирования произвольного изображения несравненно шире. Поэтому, в частности, мощной персональной ЭВМ и лазерного принтера достаточно, чтобы на их основе создать настольное издательство. Уже появились и первые цветные лазерные принтеры, но еще очень дорогие. К сожалению, отечественная промышленность пока что не освоила выпуск лазерных принтеров. Подробное описание этих устройств и принципа их действия можно найти в [17].

Более дешевые и компактные по сравнению с лазерными **струйные** принтеры применяются в тех случаях, когда за небольшую плату желательно иметь цветное изображение. Принцип действия струйного принтера: жидкая краска непрерывной очень тонкой струйкой, фактически мелкими капельками, выдавливается из емкости в сторону бумаги. Летящие капли отклоняются электрическим полем, которое управляется процессором. Нетрудно сделать несколько емкостей с красками разных цветов и тем самым обеспечить многоцветное изображение. Получается устройство, практически бесшумное, небольшое и относительно несложное. Бумага может применяться любая, писчих сортов. К недостаткам можно отнести время, необходимое для высыхания краски после завершения печатания листа, скорость печати,которая в принципе не может быть высокой. Кроме того, засыхающая краска требует периодического технического обслуживания устройства. Образец черно-белого струйного принтера показан на рис.11.

Существуют и многие другие типы печатающих устройств – электрические пишущие машинки, термографические и ременные принтеры, печатающие лазером через копирку, но все они либо уже вышли из употребления, либо неудобны и потому мало применяются, либо их серийный выпуск задерживается.

Устройства для получения векторных (штриховых) изображений на бумаге появились давно, они называются графопостроителями, или плоттерами. Современные их варианты (рис.12) имеют несколько (7-8) сменных пишущих узлов (перьев) с чернилами разных цветов, механизм

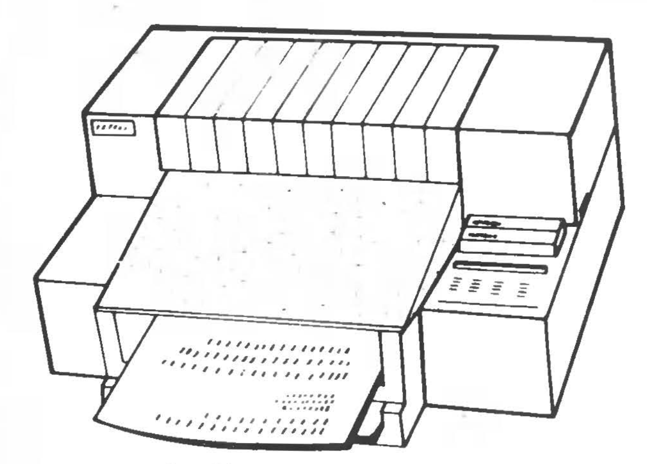


Рис. 11. Струйный принтер

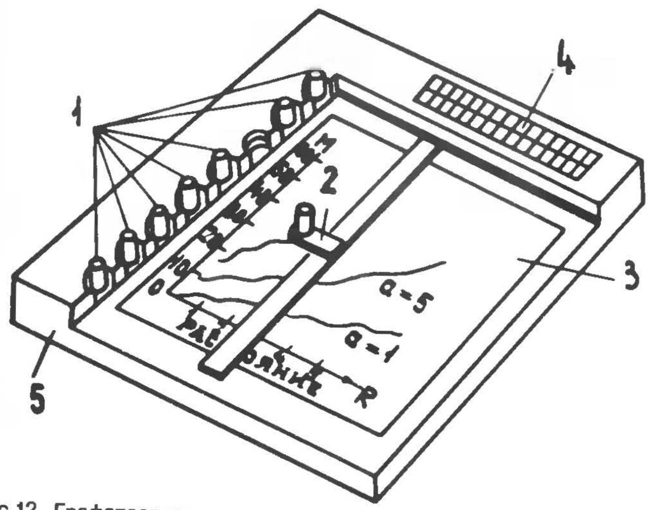


Рис.12. Графопостроитель планшетного типа: *1* – сменные пишущие узлы с цветными чернилами; *2* – подвижный рисующий механизм с одним узлом; *3* – бумага; *4* – лампочки сигнализации и клавиши управления; *5* – корпус

перемещения узла по двум координатам с минимальным шагом порядка 0,1 мм, механизм автоматической смены узла. Для вычисления координат промежуточных точек и управления отдельными механизмами к графопостроителю прилагается специальная обслуживающая программа. Некоторые графопостроители имеют собственную буферную память для временного хранения координат отрезков и микроЭВМ для вычисления координат промежуточных точек.

На рис.12 показан малогабаритный настольный вариант планшетного графопостроителя. Его можно использовать во всех случаях, когда размеры рисунка или чертежа не выходят за пределы стандартного листа бумаги. А крупноформатные чертежи получают на графопостроителе, планшет которого представляет собой массивный металлический стол.

Рулонные графопостроители отличаются тем, что в них пишущий узел движется только по одной координате, а перпендикулярное смещение выполняется многократным возвратно-поступательным движением бумаги. Такой принцип позволяет резко уменьшить габаритные размеры и вес графопостроителя за счет отказа от громоздкого стола при сохранении большого формата бумаги (рис.13). Но при этом требуется перфорированная бумага; количество цветов ограничено.

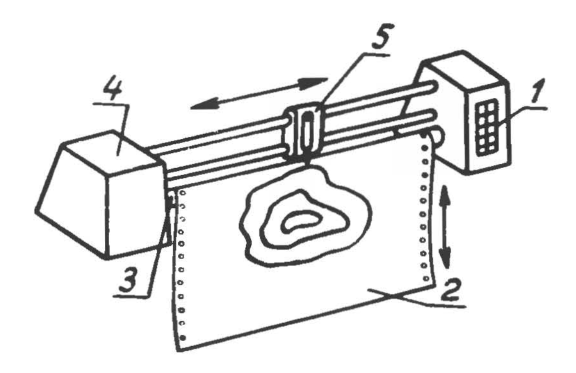


Рис.13. Графопостроитель рулонного типв: *1* – лампочки сигнализации и клавиши управления; *2* – перфорированная бумага; *3* – барабан и механизм подачи бумаги; *4* – корпус с механикой и электроникой; *5* – подвижный рисующий механизм с одним узлом

Дисплеи

Главным средством передачи информации от ЭВМ человеку в последнее время стал дисплей. Первоначально получили распространение алфавитно-цифроеые дисплеи на основе электронно-лучевых трубок (кинескопов). Они сравнительно просты по конструкции, дают довольно четкое и яркое изображение. На экране такого дисплея помещается 24-25 строк по 40...80 символов. Цвет изображения белый, зеленый или желтый. Размер точек, составляющих символы, менее 0,5 мм, поэтому получается достаточно четкая "картинка". Для уменьшения бликов поверхность экрана делают матовой. Яркость и контраст регулируются, и их всегда следует подстраивать по своему зрению и внешнему освещению. К недостаткам алфавитно-цифровых дисплеев можно отнести то, что они не позволяют получать изображение графического характера. Да и набор символов

обычно постоянен и ограничен в лучшем случае двумя регистрами двух алфавитов (как в ДВК -1 и ЕС 7927). Изменение состава или начертания символов практически невозможно.

Более современными являются графические дисплеи, позволяющие на одном и том же экране размещать одновременно тексты, рисунки, графики, схемы. Монохромные (одноцветные) графические дисплеи обычно имеют градации серого, т.е. яркость отдельных точек может устанавливаться ступенчато в некоторых пределах программным путем. Чем больше градаций, тем более близкое к фотографическому можно получить изображение (как на экране черно-белого телевизора и даже лучше).

Основным параметром монохромных дисплеев является его разрешающая способность, которая выражается в числе воспроизводимых точек. Чем больше точек, тем более четкое, с мелкими деталями изображение можно получить. В дисплеях наилучшего разрешения число точек составляет миллионы и более. В простых дешевых дисплеях обычным является число точек (640x200), равное 128000. Количестао градаций в них составляет 8...16.

Наиболее популярны в настоящее время цветные дисплеи. Они тоже используют в качестве экрана кинескоп. Одним из важнейших параметров цветных дисплеев является цветовая палитра, т.е. количество оттенков, которые можно воспроизаести путем смещения трех основных цветов. Типичная палитра содержит десятки оттенков. Число точек на цветном экране – тоже немаловажный фактор. В дисплеях профессионального применения оно составляет порядка миллиона и выше, в более дешевых, бытовых, – около 500x300.

Дисплеи на основе кинескопов характерны тем, что время свечения внутреннего покрытия экрана (люминофора) после его возбуждения пучком электронов составляет доли секунды. Если бы время послесвечения было больше, то быстро движущиеся изображения оставляли бы след, ухудшающий качество "картинки". Поэтому изображение в кинескопе необходимо постоянно обновлять с частотой десятки раз (кадров) в секунду. В бытовых телевизорах изображение обновляется непрерывно поступающим с телецентра сигналом.

В таких алфавитно-цифровых дисплеях, как ЕС 7927, имеется постоянная память знакогенератора, содержащая матрицы символов, и оперативная память, емкостью около 2 Кбайт, для поступающих от процессора кодоа символов – содержания экрана. По схеме, приведенной на рис.10, осуществляется непрерывное формирование символов пучком электронов непосредственно на экране. Буферного ОЗУ на все количество возможных светящихся точек в таких дисплеях нет. Роль программы-драйвера играет электронная схема, поскольку алгоритм размещения символов, строго определенных по форме, очень прост.

Для растровых дисплеев необходима специальная буферная память, в которой каждой точке растра соответствует несколько бит. Опрос содержимого памяти и управление лучом кинескопа осуществляет довольно

сложное электронное устройство, которое называют еще дисплейным процессором. Буферная память, имеющая также название "битовая карта", вместе с дисплейным процессором образуют графический адаптер. Он обычно представляет собой отдельную небольшую плату, которая в персональной ЭВМ вставлена в основной (системный) блок.

Графические адаптеры, а их существует много, различаются емкостью памяти и, следовательно, числом точек, которые отображаются на экране дисплея. Для получения большого числа градаций яркости в монохромных дисплеях и для управления тремя лучами также с изменениями яркости в цветных диспелях на каждую точку растра приходится до десятка и более бит. Чтобы получить изображение еще более высокого качества, необходимо увеличить частоту обновления изображения, т.е. повысить скорость дисплейного процессора.

В любом случае сохраняется приведенная на рис.10 общая схема заполнения буферной памяти (битовой карты) программой, которую выполняет основной процвссор ЭВМ, готовя по поступившей в кодах информации ее экранное изображение. А дисплейный процессор непрерывно, не мешая основному, работает только на чтение буферной памяти, постоянно обновляя изображение на экране. Структура ПЭВМ в этой части приведена на рис.14, заимствованном из [18]. В ней читатель найдет более подробные сведения о растровых дисплеях и графических адаптерах.

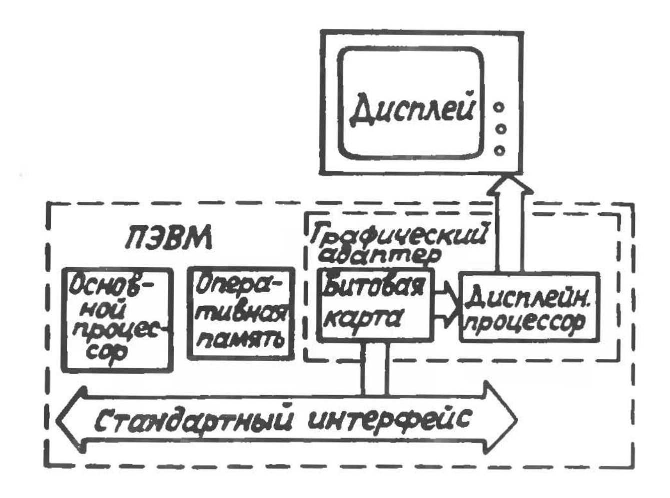


Рис. 14. Схема подключения дисплея в ПЭВМ

В переносных и портативных ("накопенных" – laptop, рис.15) ЭВМ для уменьшения веса и габаритов используются дисплеи на жидких кристаллах (как в наручных цифровых часах или во многих калькуляторах). Основу дисплея составляют две стеклянные пластины, между ними по-

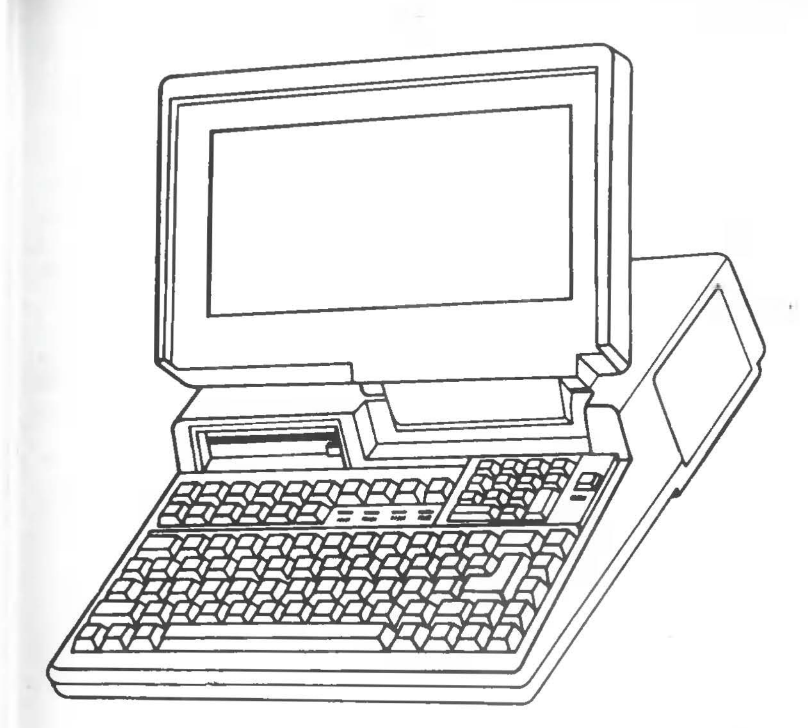


Рис. 15. Портатианая (laptop) персональная ЭВМ

мещены точечные жидкие кристаллы, напряжение к которым подводится с помощью прозрачных проводников. Под дествием электричества меняется прозрачность кристалла, и он становится видимым в отраженном свете. Сам кристалл не светится. Толщина такого дисплея не превышает нескольких миллиметров, а масса – сотен граммов. Эти дисплеи при больших размерах довольно сложны в изготовлении, поэтому их цвна еще велика.

Основным недостатком жидкокристаллических дисплеев является низкая контрастность изображения, которая к тому же падает с уменьшением освещенности. Если изображение на кинескопах в темноте лучше видно, чем при свете, то изображение на жидких кристаллах в темноте практически не видно. Зато такой дисплей потребляет очень мало электроэнергии, что немаловажно при батарейном питании в походных условиях. Другим недостатком жидкокристаллических дисплеев является их инерционность, поэтому они непригодны для быстро меняющихся изображений.

В модернизированных вариантах для повышения контраста применена подсветка. Жидкокристаллическая панель освещается с обратной стороны, и потемневшие кристаллы, образующие картинку, становятся лучше видны.

Размеры современных дисплеев позволяют размещать 25 строк по 80 символов, т.е. обеспечивается общепринятый стандарт. Легко реализуется на них и графический режим. В отечественных микрокалькуляторах последних выпусков уже применяются уменьшенные до нескольких строк жидкокристалличвские экранчики, работающие в алфавитно-цифровом и графическом режимах. В ближайшие годы дисплеи этого типа должны занять достойное место в ряду доступной для пользователя портативной техники отечественного производства после решения задачи снижения цены на большие (с диагональю 20...25 см) экраны и повышения контраста изображения.

Другой разновидностью плоских экранов являются плазменные панели. В них каждая точка – это отдельная микроскопическая неоновая лампочка. Такие дисплеи дают хорошо видимое оранжево-красное изображение с градациями яркости. Зарубежные фирмы уже выпускают портативные компьютеры с плоскими плазменными экранами.

В последние годы дисплейная технология быстро прогрессирует, поэтому можно ожидать появления в этой области новых удачных технических решений.

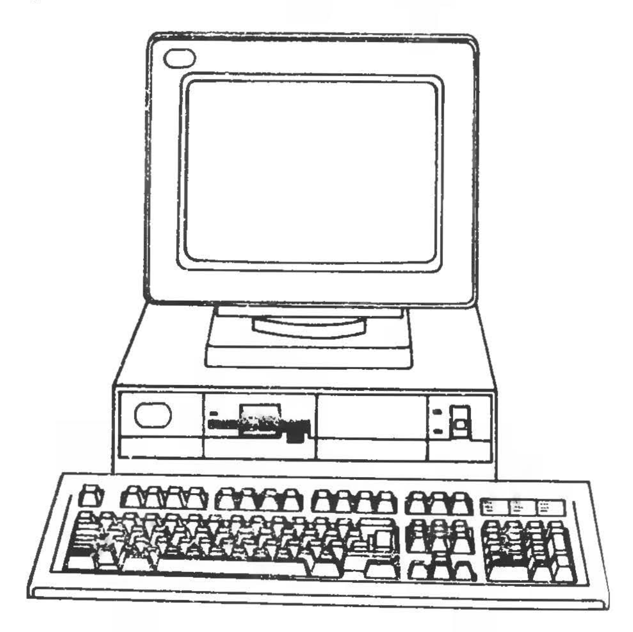


Рис. 16. Персональная ЭВМ настольного исполнения

На рис.16 показана персональная ЭВМ семейства PS/2 модель 50 Z фирмы IBM PC (США) с цветным дисплеем на кинескопе, вогнутой полноразмерной клавиатурой со 101-й профильной клавишей, дисководом для

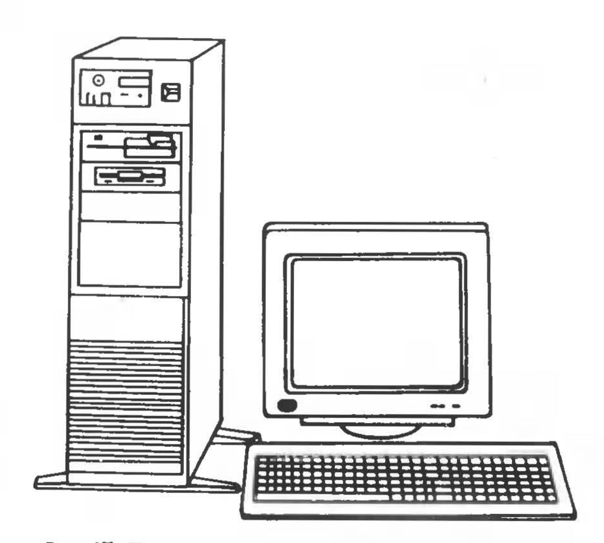


Рис. 17. Персональная ЭВМ в "башенном" корпусе

гибких дисков 3,5 дюйма. На переднюю панель выведены индикатор накопителя на жестких дисках и выключатель питания. На рис.17 показана мощная профессиональная персональная ЭВМ семейства IBM PC/AT в "башенном" корпусе с НГМД 5,25 и 3,5 дюйма.

Основные параметры ПЭВМ, аналогичных приведенным на рис.16 и 17: оперативная память 1...2 с возможностью наращивания до 16 Мбайт: основной процессор фирмы Intel 80286 или 80386 (или аналогичные им других фирм) с тактовой частотой 10...20 МГц; математический сопроцессор (процессор "плавающей" арифметики) Intel 80287 или 80387 и их аналоги; накопитель на жестком магнитном диске ("винчестер") 5,25 дюйма емкостью от 30 до 120 Мбайт; накопитель на гибком магнитном диске 5,25 дюйма емкостью 360 Кбайт и 1,2 Мбайт или НГМД 3,5 дюйма емкостью 1,44 Мбайта; видеографической адаптер (они имеют обозначения EGA и VGA) цветного дисплея, дающий до 256 цветов из палитры 256 тысяч оттенков и до 64 градаций яркости в монохромном режиме.

По желанию заказчика ПЭВМ дополнительно комплектуется (на рисунках не показаны) мозаичными принтерами, манипуляторами "мышь", накопителями на оптических дисках емкостью 200 Мбайт, кассетными накопителями на магнитной ленте и другими устройствами.

Фирмами многих стран выпускаются миллионы машин в год, свыше 1000 аналогов (клонов) ПЭВМ IBM PC, частично или полностью совместимых с ними.

Накопитель на сменном магнитном диске (НСМД)

Традиционным носителем информации, появившимся несколько позднее магнитных лент, является сменный магнитный диск. Устройства внешней памяти на сменных дисках применяются на всех крупных, средних и мини-ЭВМ. Лишь персональные и микроЭВМ оснащаются дисковыми накопителями других типов.

Сменный диск представляет собой алюминиевую пластину диаметром 35 см с ферромагнитным покрытием. Часто несколько пластин собирают в один пакет. На каждой рабочей поверхности имеются дорожки равной емкости (несколько килобайт). Одноименные дорожки на разных поверхностях пакета образуют так называемый цилиндр. Головки записи-чтения собраны в единый блок, который перемещается вдоль радиуса линейным двигателем. При использовании дорожек, входящих а один цилиндр, не требуется перемещать головки и, следовательно, тратить время на ожидание. Емкость сменного диска составляет от 2,5 Мбайт в однопластинном варианте и до сотен мегабайт в дисковом пакете. Для предохранения от пыли пакет закрывается пластмассовым футляром, который полностью или частично снимается при установке пакета в накопитель.

Накопитель на сменном магнитном диске, или дисковод, предстааляет собой ящик или напольную тумбу. На одной ЭВМ имеется, как прааило, несколько однотипных дисководов, подключенных к процессору через устройство управления и канал. Установленный в накопитель пакет дисков вращается со скоростью, равной нескольким тысячам оборотов в минуту. Такая большая скорость необходима для того, чтобы сделать минимальным время ожидания процессора при записи и считывании данных. На больших машинах при типичном для них круглосуточном режиме работы наиболее часто используемые пакеты (например, с компонентами операционной системы) вращаются непрерывно в течение недель, а иногда и месяцев без остановок. Архиа машинных носителей в виде стеллажа с десятками и сотнями пакетов дисков является непременной составляющей машинных залов вычислительных центроа.

При аварийных отключениях питания и бросках напряжения на работающих дисководах нередко портится информация или даже рабочая поверхность дисковых пластин. С целью исключения таких неприятностей содержимое постоянно используемых дисков периодически, раз в неделю или две, копируют на магнитную ленту.

В последние годы совершенствоаание устройств памяти на магнитных дисках продвигалось в нескольких направлениях: значительное увеличение емкости при прежних габаритах; многократное уменьшение размеров при сохранении емкости; сокращение среднего времени ожидания на поиск и считывание-запись информации; увеличение надежности и срока службы носителей; создание дешевого малогабаритного сменного диска, пригодного для индивидуального применения, в том числе и в быту.

Накопитель на жестком магнитном диске (НЖМД)

В последние годы накопители данного типа (их еще называют накопителями на несменных магнитных дисках, или "винчестерами", а в англоязычной литературе "hard disc") приобрели большую популярность благодаря своей значительной емкости при малых габаритных размерах. Типичный НЖМД для ПЭВМ имеет емкость 10...40 Мбайт, объем 1...2 куб.дм и массу порядка 1 кг.

Лучшие из НЖМД имеют характеристики,а несколько раз превышающие указанные (например, емкость в сотни мегабайт). Ведущие зарубежные изготовители 133-миллиметровых НЖМД в настоящее время упорно штурмуют 1000-мегабайтный рубеж.

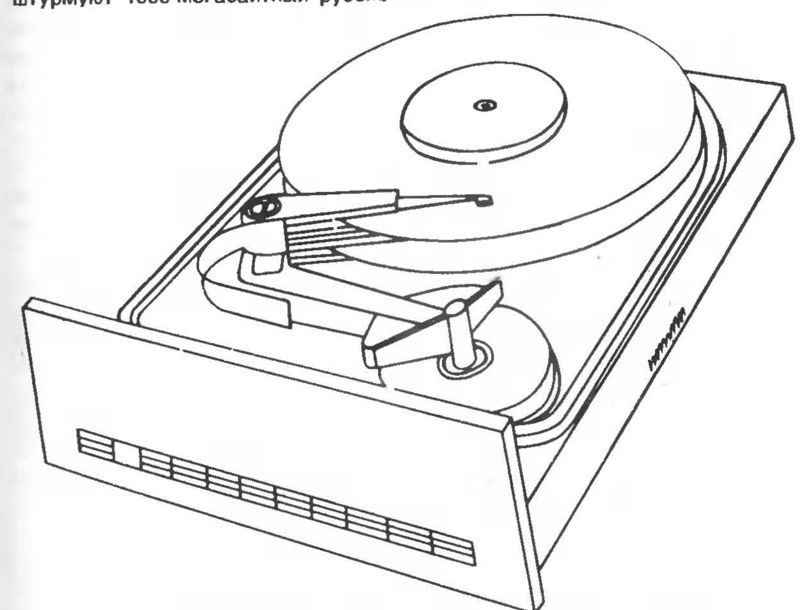


Рис. 18. Образец накопителя на жестком диске ("винчестер")

Основу НЖМД составляет паквт из нескольких дисковых пластин, покрытых с двух сторон магнитным материалом и помещенных вместе с головками записи-чтения в герметичный корпус (рис.18). На рисунке герметизирующая крышка снята, видны два диска и магнитные головки. Большинство выпускаемых НЖМД имеют диски диаметром 133 мм, а наружные габаритные размеры таких накопителей совпадают с габаритами НГМД, что очень важно с точки зрения их взаимной замены.

Для крупных ЭВМ поставляются НЖМД увеличенных размероа с повышенной емкостью а сотни и более мегабайт. В частности, ЕС 1066 комплектуется "винчестерами" емкостью 317 и 635 Мбайт, выполнвнными в виде напольной тумбы.

Для каждой магнитной поверхности дискового пакета прядусмотрена своя головка записи-чтения. Посредством держателей, соединенных с шаговым двигателем, они синхронно перемещаются по одноименным рабочим дорожкам пакета, который вращается вторым двигателем с постоянной скоростью 1,5...3,5 тыс.оборотов в минуту. Герметизация пакета и головок с помощью металлической крышки предохраняет их от пыли и влаги.

Чтобы получить необходимую емкость на сравнительно небольшой поверхности диска, ширина дорожки должна быть маленькой, менее 0,1 мм. Поэтому в накопителях повышенной емкости с большим числом дорожек для перемещения головок применяют два двигателя – один для приблизительной, а другой для точной установки на дорожку. Точная и тонкая механика НЖМД не выносит резких ударов. Перед перемещением накопителя его головки необходимо перевести в транспортное положение. В операционной системе ЭВМ для этого имеется специальная команда паркоеки головок ("park"). Некоторые современные НЖМД при выключении питания самостоятельно выполняют парковку, пользуясь запасенной энергией.

Существует около 50 логических несовместимых типов 133-миллиметровых НЖМД, отличающихся числом пластин, головок, дорожек, секторов на дорожках и их емкостью. В установочных параметрах ("setup") ЭВМ, необходимых для нормального функционирования ОС, обязательно указывается тип жесткого диска.

Двигателями и головками управляет электронный блок, размещенный на плате, прикрепленной к корпусу НЖМД и составляющей с ним одно целое. Для уменьшения габаритов блока в нем применяются микросхемы повышенной интеграции и микропроцессоры.

Аналогично дисплеям стыковка НЖМД с основным процессором ЭВМ осуществляется посредством специального блока – контроллера, или адаптера. В больших ЭВМ эта стыковочная аппаратура называется каналом. Непосредственное логическое управление НЖМД выполняет программа-драйвер, присутствующая в составе операционной системы.

Накопителями на жестких магнитных дисках (133 мм) сегодня комплектуется абсолютное большинство персональных ЭВМ, за исключением самых дешевых. НЖМД служит основным оперативным долговременным хранилещем инструментальных и пользовательских программ и данных. Внешне наличие НЖМД, расположенного, как правило, внутри основного (системного) блока машины, заметно чаще всего лишь по соответствующей сигнальной лампочке.

При включении ПЭВМ происходит раскрутка дискового пакета внутри накопителя, сопровождаемая характерным нарастающим звуком. Слабый шум от вращения пакета слышен все время, пока включена машина. Перемещение головок между дорожками легко распознать по дополнительному звуку щелчков, производимых шаговым двигателем накопителя. Отклонения от обычной звуковой картины работы ПЭВМ должны настораживать пользователя, так как они могут свидетельствовать о появлении неисправностей, требующих немедленного вмешательства.

Гибкяй диск

Дешевым, компактным и достаточно удобным носителем информации является гибкий диск (дискета). Он изготавливается из тонкой пластиковой пленки с магнитным покрытием. Существует много типов и размеров гибких дисков, но фактическими стандартами на сегодня стали два – 5,25 дюйма (133 мм) и 3,5 дюйма (89 мм). Более крупные 8-дюймоеые диски выходят из употребления.

Пользователь любой ПЭВМ, за исключением разве что простейшей, неизбежно соприкасается с гибкими дисками. Поэтому рассмотрим подробно конструкции гибких дисков указанных наиболее популярных размеров – 5,25 и 3,5 дюйма. Дискета 5.25 дюйма показана на рис.19.

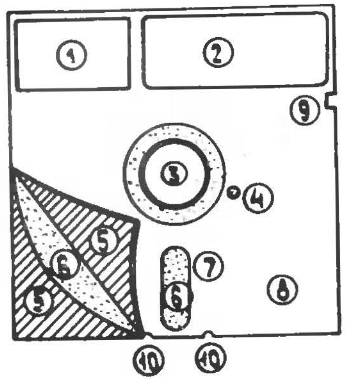


Рис.19. Конструкция мини-дискеты (5,25 дюйма): *1* – этикетка фирмы-изготовителя; *2* – заменяемая чистая этикетка; *3* – сквозное отверстие для шпиндельного привода; *4* – индексное отверстие; 5 – нетканый материал; *6* – гибкий диск с магнитным покрытием; *7* – вырез для головки записи–чтения; *8* – пластиковый защитный корпус; *9* – вырез защиты информации; *10* – освобождающие выемки; габаритные размеры дискеты 134x134x1,5 мм

Диск имеет большое центральное отверстие, через которое он приводится во вращение, и маленькое боковое, необходимое для определения начала дорожки. В некоторых дискетах для уееличения срока службы диска центральное отверстие делвется с усилительным кольцом. На поверхности диска с одной или с двух сторон располагаются концентрические дорожки. Количестео дорожек обычно раено 40 или 80.

Диск запечатан в корпус из пластика, напомчнающего тонкий картон, с необходимыми прорезями. Внутренняя стенка корпуса выполнена из ворсистого нетканого материала, который уменьшает трение между диском и корпусом и поглощает мелкую пыль, попадающую внутрь дискеты, не давая ей царапать поверхность диска. Общая толщина диска и корпуса не превышает 2 мм. На корпусе имеются этикетка изготовителя и свободное место для пометок. Для предохранения от пыли и загрязнений дискета хранится в бумажном конверте. Диск из корпуса никогда не вынимается. На краю дискеты имеется прямоугольный вырез, служащий

для защиты от случайной записи. Если этот вырез заклеить (в коробке с дискетами припагается липкая фольга), то запись на та́кую дискету или стирание станут невозможными.

На этикетке дискеты указывается, какая плотность записи гарантируется и сколько сторон рекомендуется использовать: по числу сторон – одна или две, а по плотности записи – одинарная, двойная и повышенная. Видимая в прорези поверхность гибкого диска должна быть матовой и почти черной. На изношенной дискете в длинной прорези можно увидеть дорожки (точнее, царапины от пыли и контакта с головкой).

Дискета без бумажного конверта вставляется в дисковод, длинной прорезью вперед (от себя), этикеткой вверх (при горизонтальной щели). После вставки дискеты нужно закрыть замок или повернуть защелку в зависимости от конструкции накопителя. Во время записи или чтения информации диск вращается в гибком корпусе со скоростью 300 оборотов в минуту с характерным тихим шелестом или шипением. При перемещении магнитных головок слышны щелчки. Во время работы дисковода на нем светится лампочка. Если процессор долго не обращается к диску, вращение прекращается, и лампочка гаснет. Брать дискету следует осторожно за углы, не гнуть и не мять, не касаться самого диска, хранить при комнатной температуре в вертикальном положении.

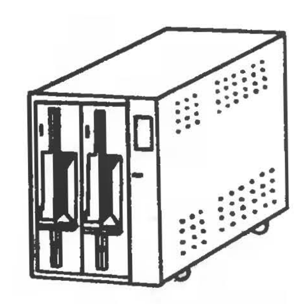


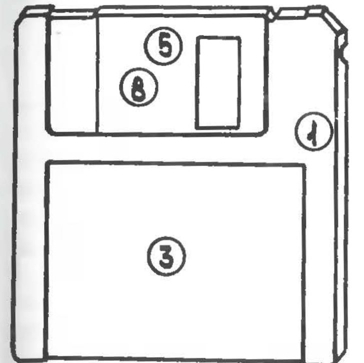
Рис.20. Накопитель на гибком магнитном диске диаметром 133 мм, сдвоенный, в отдельном корпусе

Устройство накопителя на гибком магнитном диске (рис.20) такое же, как и у больших носителвй на сменных магнитных дисках. Отличия заключаются в количестве головок, мощности двигателай в габаритных размврах устройств.

Ведущие фирмы – изготовители дискет и накопителей – интенсивно работают над их совершенствованием, поэтому можно ожидать появлвния в ближайшее время дисквт емкостью в десятки мегабайт.

Как уже говорилось, международным и отечественным стандартом фактически являются дискеты размером 133 мм (5-дюймовые), емкостью 360...1200 Кбайт в зависимости от числа используемых дорожек и качества магнитного слоя. Дискеты такого размера применяются в персональных ЭВМ. Большие дискеты (8-дюймовые) используются только в мини-ЭВМ. В последнее время используются микродискеты размером

89 мм и менее. Для предохранения дисков от порчи в таких дискетах используется жесткий корпус (тонкая пластмассовая коробочка). Емкость 89-миллиметровых микродискет (рис.21) составляет 720 или 1400 Кбайт.



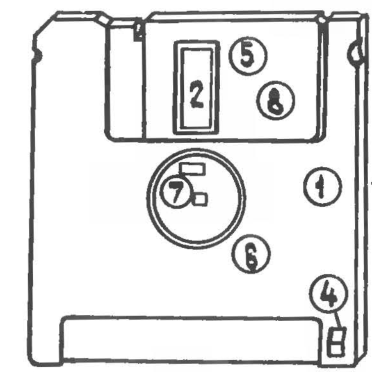


Рис.21. Конструкция микродискеты (3,5 дюйма): *1* – полужесткий пластмассовый корпус; *2* – гибкий магнитный диск; *3* – место для чистой этикетки; *4* – переключатель блокировки (защиты) записи; *5* – отверстия для головок записи–чтения; *6* – металличаский сердечник; *7* – отверстия в металлическом сердечнике для центровки и вращения; *8* – металлический замок отверстия для головок (на правом рисунке – открыт, поверхность диска доступна); левый рисунок – передняя сторона; первый – задняя; габаритные размеры дискеты 90x90x3 мм

Электронные квазидиски

При интенсивном общении пользователя ПЭВМ с файлами на дисках становятся ощутимыми затраты времвни на обмен с диском – поиск нужной дорожки по каталогу, подвод к ней головок, собственно считывание или запись. Сократить эти потери можно, если резерв оперативной памяти составляет 200...500 Кбайт. В условиях устойчивого электроснабжения очвнь удобно пользоваться электронным квазидиском, потому что он работает в десятки раз быстрее жвсткого и гибкого дисков и совершенно бесшумно.

Создание электронного квазидиска, который еще называют виртуальным диском, возможно несколькими способами. Пврвый, наибопее простой – использоаать часть ОЗУ с помощью программы-драйвера электронного диска. Второй – подключить специальную электронную память (отдельную плату или целый блок), имитирующую запоминающие свойства и операции обычного диска (гибкого или жесткого), но с повышенной скоростью. При работе с квазидиском необходимо периодически, примерно каждый час, запоминать редактируемые файлы на настоящем диске, поскольку а случае сбоя ЭВМ или отключения элвктроэнергии содержимое ОЗУ безвозвратно теряется.

В качестве примера полезного применения электронного диска можно назвать ситуацию, когда пользоватвлю необходимо неким сложным об-

разом перекомпоновать несколько не слишком длинных файлов. Их можно временно разместить на электронном диске, тогда многочисленные переносы кусков текста из одного файла в другой займут заметно меньше времени.

Оптические диски

Перспективными устройствами памяти, особенно большой емкости, являются оптические диски. За счет применения тонко сфокусированного луча, испускаемого миниатюрным полупроводниковым лазером (лазерным диодом), удается значительно уменьшить площадь, занимаемую на поверхности диска каждым битом информации. Емкость оптического диска самое меньшее в десять раз превышает емкость диска того же размера с магнитным диском и составляет сотни мегабайт.

Широкому распространению оптических дисков препятствует неотработанность технологии многократной записи-стирания. Наиболее популярные и дешевые оптические диски уже на этапе изготовления наполняются какой-то определенной информацией, и их невозможно переписать. Существуют более дорогие диски однократной записи, на которые один раз можно занести любые данные, но затем их удается только считывать. Поэтому в настоящее время оптические диски применяются для хранения относительно постоянных сведений справочно-энциклопедического характера.

Есть и еще один недостаток у оптических дисков, не устраненный на сегодняшний день. Он заключается в длительном времени поиска нужных данных. Среднее время поиска составляет секунды, в то время как даже в НГМД среднее время позиционирования головок не превышает долей секунды. Большое время доступа у оптических дисков получается из-за необходимости просматривать большое число дорожек и анализировать большой объем информации, поэтому для улучшения характеристик доступа требуется усовершенствовать электромеханическую часть и повысить быстродействие электронных компонентов оптического накопителя.

Размещение информации на магнитном диске. Как уже отмечалось, вся поверхность диска разделена на концентрические дорожки, которые, в свою очередь, делятся на блоки или сектора. Разбивка диска на эти мелкие участки, проверка их пригодности для хранения информации, отбраковка плохих участков осуществляется на этапе подготовки нового диска к эксплуатации. Иногда такая подготовка делается фирмой-изготовителем, а чаще – самим пользователем с помощью программ форматирования, входящих в состав операционной системы.

Число дорожек и секторов, длина сектора в байтах зависят от конструкции накопителя и режима форматирования. Как правило, существует возможность отформатировать один и тот же носитель на несколько емкостей, меняя число дорожек, секторов и объем сектора. Так, двусторонние 133-миллиметровые дискеты двойной плотности (DD – double density) можно отформатировать на 360 К, т.е. одинарную плотность, при этом на каждой ее стороне будет заведено по 40 дорожек, содержащих по 9 секторов размером а 512 байт каждый (это наиболее совместимый формат, поддерживаемый на всех ПЭВМ – аналогах IBM PC.) или на 720 К за счет удвоения числа дорожек. А если позволяют конструкция накопителя и качество магнитного слоя, емкость дискеты можно довести до 1200 К, увеличив до 16 число секторов на дорожке.

В процессе форматирования на нулевую дорожку записываются сведения о плохих секторах. В дальнейшем они не участвуют в хранении информации. Если же сама нулевая дорожка окажется с дефектом, то такая дискета для работы не годится. На нулевой дорожке ОС формирует каталог диска, занося в него имена файлов, их размеры и даты создания, сведения о занятых секторах, их начальных адресах, номера свободных секторов. По каталогу диска осуществляются поиск и считывание данных.

Накопители на магнитной ленте (НМЛ)

Запоминающие устройства на магнитной ленте являются традиционными в перечне средств вычислительной техники. В состав средних и больших ЭВМ НМЛ входят обязательно и в нескольких экземплярах. Это объясняется дешевизной и сравнительно большой емкостью носителя. Накопители на магнитной ленте выпускаются во множестве модификаций – от громоздких шкафов до миниатюрных коробочек, умещающихся на ладони. Многие радиолюбители используют бытовые магнитофоны в качестве накопителя информации для персонального компьютера.

Основным недостатком магнитной ленты как носителя данных является большое среднее время доступа, т.е. велико время поиска и подвода к магнитной головке нужного участка носителя информации. Чтобы найти какой-то текст, расположенный на другом конце ленты, ее нужно всю перемотать. При большой длине ленты (например, на ЕС ЭВМ) на это может тратиться более 10 минут.

Наиболее распространенным стандартом в нашей стране является магнитная лента шириной 12,7 мм (0,5 дюйма), которая широко применяется на машинах семейств "Эльбрус", ЕС и СМ. На ленте формируется 9 дорожек, из которых 8 являются информационными, а девятая – контрольной. По ширине ленты помещается ровно один байт. Продольная плотность записи зависит от качества ферромагнитного слоя и составляет 8 (пониженная), 32, 63 (повышенная) и 246 бит на миллиметр длины дорожки. Чаще всего применяется плотность 32 бит/мм.

В последние годы в рамках ЕС ЭВМ появились накопители с плотностью записи 63 бит/мм. Значительного повышения плотности удалось добиться ценой отказа от старт-стопного режима движения и перехода к режиму "бегущей" (stream) ленты. В этом режиме лента не останавливается на каждом блоке данных, а движется непрерывно с постоянной скоростью. Такой режим хорошо согласуется с "выгрузкой" содержимого диска большой емкости, поэтому НМЛ с бегущей лентой применяют для защиты информации, расположенной на жестких дисках.

Накопители на кассетной магнитной ленте находят широкое применение лишь в простейших бытовых ПЭВМ в качестве осноаной анешней памяти при полном отсутствии дискоаых накопителей. Кассетные накопители старт-стопного режимв и режима бегущей ленты применяются в ПЭВМ для разгрузки жестких дисков. Емкость компакт-кассеты а таких НМЛ составляет десятки мегабайт. В базовую конфигурацию персональной ЭВМ накопители на магнитной ленте, как правило, не входят.

Оптические сканеры

Ввод первичной информации с различных документов а память машины чаще асего производится вручную посредстаом клааиатуры. Этот низкопроизаодительный, не требующий особой квалификации труд можно заменить на более эффективный путем применения специальных устройств, назыааемых сканерами, которые позволяют считывать любые тексты (кроме рукописных) с хорошей скоростью порядка сотен символов а секунду.

Некоторые сканеры своим внешним аидом напоминают фотоувеличитель: над небольшим столиком, на стойке, укреплен оптический прибор наподобие миниатюрной телеаизионной камеры. В комплект сканера входит программа-анализатор, аыделяющая на поле из саетлых и темных точек образы бука, цифр и других символов и передающая а память ЭВМ их коды. Такой анализатор а принципе можно "научить" распознааать любой шрифт любого алфааита. Практически достаточно, чтобы сканер мог читать одновременно 3-4 шрифта на одном или дaух языках (алфавитах). Соаременные варианты сканеров приспособлены к "обучению" незнакомым шрифтам прямо а процессе работы. Для этого достаточно ввести с клааиатуры символы, соответствущие зрительным образам, "уаиденным" анализатором при "чтении" первых бука текста.

Зарубежными фирмами выпускаются разные типы сканеров, в том числе малогабаритные, по форме похожие на "мышь". Таким сканером нужно аодить по считыааемому тексту. Скорость ввода печатного текста состаалает десятки симаолов а секунду.

Следует подчеркнуть, что речь пока идет о чтении именно печатного текста, имеющего в пределах документа фиксированные аид и размер. Исследования и разработки по распознааанию рукописного текста еще находятся а стадии зкспериментов. Достигнутый процент (80...90) прааильного узнааания даже при аккуратном почерке не позволяет считать такое качестао приемлемым для массового повседнеаного применения. Кроме того, сама потребность во ааоде рукописного текста со аременем должна сокращаться по мере оснащения произаодителей информации (проща гоаоря, всех пишущих людей) персональными ЭВМ и терминалами больших машин.

Кроме чтения печатного текста сканеры обеспечиаают ввод растровых изображений – рисункоа, фотографий, картин (рис.22). Для сканероа это более простой режим, чем ввод и распознааание текста.

Рис. 22. Настольный сканер

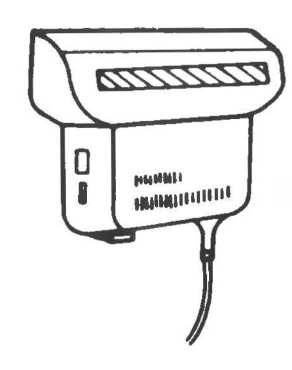


Рис. 23. Ручной сканер

Ручной сканер ("handy scanner", рис.23) охватывает полосу чтения, шириной 10,5 см, максимальное разрешение, зааисящее от скорости движения сканера, составляет от 4 до 16 точек на миллиметр пути. Требуемый минимальный объем оперативной памяти состаалает около 400 Кбайт. Программа, обслужиааюющая сканер, позаоляет перемещать, поворачивать, масштабироаать по любой координате и удалять фрагменты изображения, накладыаать на изображение текст, комбинировать фрагменты нескольких изображений, инвертироаать картинку (переводить позитиа в негатиа и наоборот) и модифицировать ее с помощью "мыши", записыаать на диск и считыаать с него. Оптические сканеры так же, как и аысококачестаенные принтеры, станоаятся непременной принадлежностью настольных издательств.

Манипулятор "мышь"

Срaанительно новыми периферийными устройствами, появившимися впераые в составе персональных ЭВМ, являются специальные манипуляторы типа "мышь" ("mouse", рис.24), "ручка" ("джойстик", "joystick", рис.25) и "шар". Они предназначены для перемещения курсора на дисплеях, имеющих графический режим работы, по любой траектории и с любой нужной скоростью. Обычные клавиши-стрелки, позволяющие двигать курсор а любую позицию, т.е. по горизонтали, любой строки, т.е. по вертикали, алфавитно-цифрового дисплея, не годятся для графического изображения. Если клавиша-стрелка за одно нажатиа сдвигает курсор только в вертикальном или горизонтальном направлении на одну точку экрана, то сколько потребуется нажатий для перемещения на 300 точек по диагонали ? А если курсор "бежит" при нажатой клавише, то как менять его скорость? Удовлетаорительным является аатоматическое плавное нарастание скорости при длительном нажатии клавиши-стрелки. Но и это не совсем то, что нужно.

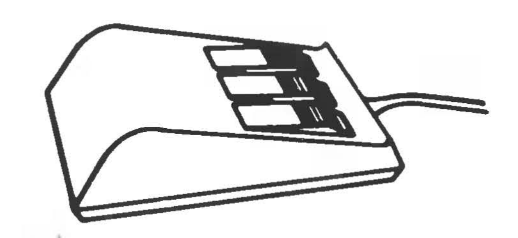


Рис.24. Манипулятор "мышь" с тремя кнопками

Рис.25. Манипулятор "джойстик" с курком и кнопкой для компьютерных игр

Наиболее удачным техническим решением явилось принципиально ноеое устройство упраепения курсором, которое за сеое енешнее сходство получило название "мышь". Суть его действия заключается в том, что курсор на экране повторяет есе перемещения, которые пользователь делает, даигая "мышь" по столу или любой другой гладкой поверхности. Сам курсор графического режима формируется программой и часто представляет собой стрелку, острие которой указыеает как раз на рассматриваемую точку.

Главной деталью "мыши" является небольшой шар, вращение которого в разные стороны отслежиеается специальной системой, передающей изменения координат при перемещении манипулятора. Для предотвращения проскальзыеания шар утяжелен и покрыт резиной, а в комплекте с "мышью" имеется шероховатая подставочка. Скорость перемещения "мыши" может быть любой, коэффициент передачи (отношение длины пути "мыши" к длине пути курсора) легко регулируется изменением одного из параметров, задаваемых в драйеере "мыши".

На корпусе "мыши" имеются две или три кнопки, по своему назначению аналогичные наиболее часто употребляемым клавишам исполнения (таким, кек "enter"). "Мышь" подсоединяется либо к клавиатуре, имеющей

разъем и встроенный адаптер, либо к основному блоку ПЭВМ также через адаптер. Для работы с "мышью" необходим особый программный драйвер, соотеетствующий ее конструкции.

Многие соеременные программные средства, рассчитанные на массоаого пользоаателя, приспособлены к упраелению посредстеом меню и "мыши". В алфаеитно-цифроеом режиме работы дисплея курсор "мыши" представляет собой цветной прямоугольник, который дискретно, прыгая с симеола на символ и со строки на строку, движется по экрану.

Манипуляторы "ручка" и "шар" по своему назначению ничем не отличаются от "мыши". В них лишь несколько иной способ преобразования движения руки человека в перемещение курсора. В сфере профессионального применения ПЭВМ "ручка" и "шар" используются редко.

Кодирующий планшет

Распространенными средстеами ввода таких графических данных, как чертежи, схемы, планы, являются кодирующие планшеты (дигитайзеры, сколки). Планшет состоит (рис.26) из плоской панели и чуаствительного элемента – визира, е который встроены лупа и несколько кнопок. В центре лупы горизонтальной и еертикальной линиями нанесено перекрестие.

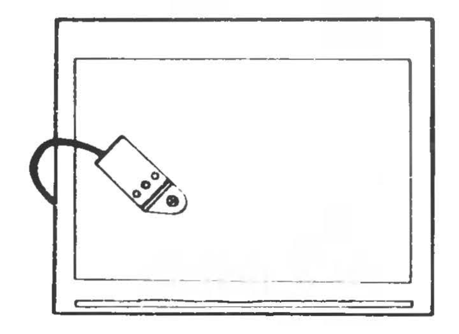


Рис. 26. Кодирующий планшет (дигитайзер)

Визир соединяется с панелью кабелем. Панель имеет роеную поеерхность для чертежа, под которой размещены электронные датчики, определяющие положение визира на поеерхности с ошибкой не более 0,1 мм. Процесс ввода данных состоит в том, что перекрестие лупы соемещают с точкой излома или пересечения линий и нажимают кнопку на еизире. Координаты точки передаются через программу-драйвер е память ЭВМ. Последовательно обходя такие характерные точки, вводят набор координат, полностью определяющих чертеж. Этот процесс является обратным процессу вывода векторного изображения. В другом варианте планшета имеется чуествительный элемент е виде карандаша, острием которого касаются точек чертежа.

АЛГОРИТМИЗАЦИЯ И ЗАПИСЬ ПРОГРАММ

Ознакомившись в предыдущих разделах с техничвской базой, в этом и следующих двух разберем основные элемвнты процесса программирования и решвния задач на ЭВМ. Традиционно он включает в свбя следующее: постановку задачи, алгоритмизацию, запись алгоритма на языке программирования (кодирование), ввод исходной программы в машину, отладку, трансляцию и исполнение. По ходу дела читатель получит представление о текстовых редакторах, причем более широкое, чем это требуется при работе с программой. Кроме того, здесь же будет дано сравнение основных конструкций нескольких популярных языков программирования.

На начальном этапе – постановки задачи – мы долго задерживаться не будем, так как для мелких простых задач на практике обычно никакой формализации не проводят, а за сложные задачи начинающему программисту браться не следует. Формулируя задачу, следует добиваться полноты, непротиворечивости, однозначности и выполнимости ее условий. Необходимо помнить, что машина – это быстрый, аккуратный, точный и вместе с тем совершенно тупой исполнитель: нельзя рассчитывать на то, что при решении задачи ЭВМ может о чем-то "догадаться", что-то "сообразить" и т.д. Соображать приходится человеку при создании программы. Из остальных этапов наиболее сложными в творческом отношении яаляются алгоритмизация и отладка программы.

Основная трудность перехода от постановки задачи к детальному алгоритму (алгоритмизации) заключается в том, чтобы мысленно представить и записать без ошибок асю последовательность действий машины, которые необходимы для получения правильного решения. Опыт автора показывает, что удобным вспомогательным приемом при переходе от постановки задачи к алгоритму является наглядное схематическое изображение процесса преобразований (перемещений, перестановок и т.д.) объектов, входящих в описанную в задаче ситуацию. Такое наглядное разъяснение "для себя" позволяет избежать многочисленных ошибок, которые всегда появляются при алгоритмизации сколько-нибудь сложной задачи "в уме". Использование известного метода блок-схем рекомендуют далеко не все специалисты, поскольку блок-схемы довольно абстрактны и мало помогают наиболее сложному творческому процессу – формированию основной идеи алгоритма решения. Более удобны в этом отношении, как мы считаем, рисунки, на которых изображаются: а) основные объекты (элементы) задачи; б) их обозначения в будущей программе; в) действия по изменению и перемещению объектов; г) начальное, промежуточные и конечное состояния (расположения) объектов.

Поясним сказанное простым примером. Возьмем известную задачу о волке, козе и капусте, которых человек с помощью маленькой лодки должен перевезти с одного берега реки на другой, не допустив при этом, чтобы коза съела капусту или волк – козу. Обозначим движущуюся лодку с человеком стрелкой влево или вправо, а волка, козу и капусту – соот-

ввтствнно В, КЗ, КП. Исходное состояние задачи: все на лввом бервгу. Первов двйствив: в лодкв везут козу. Нетрудно догадаться, что в двух остальных вариантах начала выполнения (остаются волк с козой или коза с капустой) кто-то кого-то съедает, т.е. будет нарушено одно из условий задачи. Состояния, когда лодка находится у берега, будет показывать в строках одно под другим. Между строками состояний изобразим стрелку-действие с перевозимым объектом.

Схема работы			Комментарий
В,КЗ,КП			начальное положение; везти можно только козу
	КЗ - - →		
В,КП		КЗ	после первого перевоза
	← - -		
В,КП		КЗ	после обратного "пустого" рейса
	КП - - →		везет капусту
В		КЗ,КП	оставлять козу с капустой нельзя!
	КЗ ← - -		козу обратно
В,КЗ		КП	теперь нужно волка везти к капусте
	В - - →		
КЗ		КП,В	осталось съездить за козой
	← - -		
КЗ	КЗ - - →	КП,В,КЗ	задача решена

Из схемы видны все необходимые действия (их семь), не нарушающие поставленных условий, которые следуют по логике из начального состояния и всех промежуточных. Конечно, требуется некоторое терпение, чтобы изобразить ВСЕ промежуточные состояния и ВСЕ нужные действия. Но, глядя на подобную схему, гораздо легче без ошибок написать программу. Заметим также,что на рисунке предельно упрощенно показаны лодка и человек, поскольку для поиска решения задачи их действия по перевозке достаточно изобразить стрелкой.

Следует обратить внимание на одну важную деталь. Комментарий, помещенный справа, – это почти готовый алгоритм решения задачи, записанный на обычном языке. Если его немного поправить, точно обозначая КАЖДОЕ действие, то получится запись, которую назыаают псевдокодом.

Построение алгоритма можно начинать с рисунка, псевдокода или того и другого вместе. Хотелось бы только подчеркнуть, что псевдокод еще менее нагляден, чвм блок-схема, да и полностью первписывать его при изменении алгоритма – занятие утомительное.

Разберем еще один простой, но более реальный пример. Пусть имеется последовательность каких-то однородных элементов (чисел, букв и т. п.): R[1], R[2], R[3],...,R[N]. Необходимо переставить элементы в обратном порядке, используя как можно меньше памяти (т.е. вспомогательных переменных), – R[N], R[N–1],...,R[3], R[2], R[1]. Для начала заметим, что если N – нечетное, то один элемент – центральный – никуда переставлять не нужно, он все равно окажется на том же самом месте. При четном N надо переставлять все элементы. Возьмем в качестае последовательностей слова КНИГА и КЛАД. Изобразим оба (четный и нечетный) варианта решения задачи (рис.27). Цифры в кружках – это порядкоаые номера действий. Клетки с буквами – ячейки (точнее, байты) памяти. Напомним, что в памяти ЭВМ хранятся не сами изображения букв, а их коды.

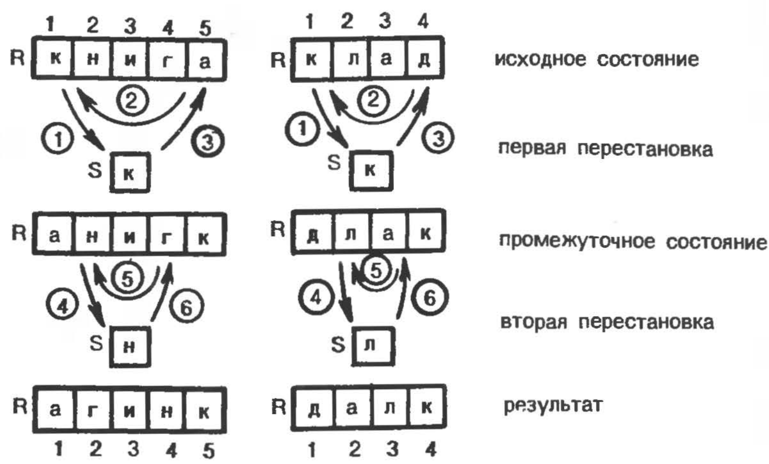


Рис.27. Схема перестановки последовательностей бука в обратном порядке

Здесь показаны два варианта массива: R [1:5] и R [1:4]. В качестве промежуточной (буферной переменной) используется переменная S. Из рис.27 видно, что надо сделать всего 2 перестановки за 6 действий при обьемах массива 5 и 4 элемента, т.е. если размер массива равен N, то число перестановок P можно получить, поделив N на 2 и отбросив дробную часть. Во многих языках есть функция ENTIER:

P: = ENTIER (N / 2 + 0.1) ;

Величина 0.1 добавлена для того, чтобы после деления целого четного на 2 получившеася вещественное значение, немного больше или немного меньше, чем точное целое частное, стало зааедомо больше этого целого. Тогда отбрасыаанием дробной части при любом положительном N асегда будет получаться правильный результат. Этого же эффекта можно достичь,

если использовать операцию деления без остатка. Здесь она не употреблена, потому что в разных языках обозначается неодинаково.

Составим таблицу индексов для чётных и нечётных случаев. Теперь не составит труда, глядя на рис.27 и в таблицу, записать операторы перестановок для нечетного и четного случаев.

Номер литеры, копируемой в буфер	Номер замещающей литеры	
	нечетный	четный
1	5 (N)	4 (N)
2	4 (N–1)	3 (N – 1)

Нечетный: S := R [1]; R [1] := R [5]; R [5] := S; (первая)
S := R [2]; R [2] := R [4]; R [4] := S; (вторая)

Четный: S := R [1]; R [1] := R [4]; R [4] := S; (переая)
S := R [2]; R [2] := R [3]; R [3] := S; (вторая)

Четный и нечетный случай ничем не будут отличаться, если вместо максимальных значений (4 и 5) индексов подставить переменную N:

S := R [1]; R[1] := R [N]; R [N] := S; (первая)
S := R [2]; R [2] := R [N – 1]; R [N – 1] := S; (аторая)

Если массив имеет больший размер, то понадобятся еще перестановки:

S := R [3]; R [3] := R [N – 2]; R [N – 2] := S; ...

Нетрудно догадаться, что все перестановки будут выглядеть одинаково, если вместо конкретных номеров подставить переменную I, пробегающую в цикле значения 1,2,3,...,P:

S : = R[I]; R[I] : = R[N – I + 1]; R[N – I + 1] : = S;

Таким образом, окончательно алгоритм перестановки элементов одномерного массива в обратном порядке имеет вид:

```
       P := ENTIER (N/2 + 0.1); I := 0;
МЕТКА: I := I + 1; S := R [I]; R[I] := R [N – I + 1]; R [N – I + 1] := S;
       IF I < P THEN GOTO МЕТКА;
```

Можно легко убедиться, что алгоритм работоспособен при любом N > 1. Он годится также для перестановки массива, содержащего элементы любого типа, а не только символы.

Завершая обсуждение этого примера, отметим факт использования еще одного полезного средства – таблица номеров участвующих (переставляемых) элементоа. Занесенные в таблицу числа далее в алгоритме выступают уже как индексы элементоа массива.

Рассмотрим теперь более сложный и интересный пример. Пусть в некоторой строке символов имеются слова, разделенные пробелами. Строка взята из текста, в котором надо выровнять правый край. Иначе говоря, надо раздвинуть слова так, чтобы не осталось пробелов в конце строки. Допустим, что переносить буквы можно только по одной. Удобно представить строку одномерным массивом символов:

н	о	в	ы	е		Э	В	М		п	р	и	г	о	д	н	ы		д	л	я		в	с	е	х	␣	␣

Дадим массиву имя S, текущий (любой) элемент обозначим S[I]. В строке пусть будет N элементов (символов, включая пробелы), N используем для общности алгоритма. Подставим к исходному состоянию обозначения и наметим первые действия:

N – 1 N

1 2 3 4 . . .

н	о	в	ы	е		Э	В	М		п	р	и	г	о	д	н	ы		д	л	я		в	с	е	х	␣	␣

④③②①

Буква	До	После
х	N–P(–0)	N(–0)
е	N–P–1	N–1
с	N–P–2	N–2
в	N–P–3	N–3

Цифры в кружках рядом со стрелками, как обычно, означают номера соответствующих действий. Запишем в таблицу номера позиций букв слова "всех" до перестановки и после нее. Используем для общности алгоритма вспомогательную переменную P, обозначающую число пробелов в "хвосте" строки. Ясно, что до начала перестановок букв нужно определить P по содержимому строки. Для этого можно в цикле просмотреть символы с ее конца и определить номер первого, отличающегося от пробела. Затем из длины строки вычесть этот номер. В качестве текущего индекса возьмем переменную I:

```
         I := N + 1;
МЕТКА:   I := I – 1;  IF I = 0 THEN GOTO МЕТКА 2;
         IF S [I] = " "  THEN GOTO МЕТКА;
МЕТКА 2:  P := N – I;
```

Здесь мы попутно предусмотрели, чтобы алгоритм правильно работал и в крайних случаях – когда в конце строки нет ни одного пробела (P станет равным 0) и когда вся строка состоит из одних пробелов (нет ни одного слова, P = N).

Пользуясь таблицей, запишем алгоритм побуквенного копирования слова "всех" в конец строки. Обратим внимание, что брать буквы надо по порядку, начиная с последней.

```
S [N] := S [N – P];   (или для общности S [N – 0] := S [N – P – 0];)
S [N – 1] := S [N – P – 1];
S [N – 2] := S [N – P – 2];
S [N – 3] := S [N – P – 3];
```

По существу, мы переписали содержимое таблицы, добавив имя массива, квадратные скобки и символ оператора присваивания. Свернем эти четыре оператора в цикл, организованный переходом по метке, используя тот факт, что вычитаемые в индексном выражении числовые значения идут подряд: 0, 1, 2, 3,... Пусть это будет целая переменная J. В качестве знака "не равно" используем сочетание "/=", что более наглядно, чем принятое в Паскале "<>".

```
         J := – 1;
МЕТКА 3:  J := J + 1;  S [N – J] := S [N – P – J];
         IF  S [N – P – J – 1] /= " "  THEN GOTO МЕТКА 3;
```

Здесь дополнительно в последнем операторе предусмотрено, что длина слова может быть любая, а не только 4 буквы. Изобразим то, что получится после выполнения записанных операторов, и наметим стрелками следующие действия.

N – 1 N

н	о	в	ы	е		Э	В	М		п	р	и	г	о	д	н	ы		д	л	я		в	с	в	с	е	х

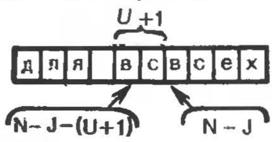
③②①

Запомним также, что к этому моменту P = 2, J = 3. Индекс N – P – J указывает на первую букву последнего слова до его перемещения. Теперь нужно определить число пробелов, которые надо поместить перед ним. В общем случае это на один пробел больше, чем было. Таким образом, надо сосчитать число пробелов между последними словами (до перестановок) и добавить единицу. Это легко сделать, если сравнивать с пробелом символы к началу строки, начиная с позиции N – P – J – 2 (в последнем алгоритме попутно определяется, что в позиции N – P – J – 1 находится пробел). Введем для обозначения числа пробелов между словами до перестановок переменную U. Алгоритм вычисления U отличается от алгоритма определения P незначительно:

```
         I := N – P – J – 1;
МЕТКА 4:  I := I – 1;  IF I = 0 THEN GOTO МЕТКА 5;
         IF S [I] = " " THEN GOTO МЕТКА 4;
МЕТКА 5:  U := N – P – J – 1 – I;
```

Теперь нужно U + 1 пробелов поместить перед сдвинутым последним

словом, которое начинается в позиции N − J, т.е. в цикле с позиции N − J − 1 к началу строки занести U + 1 пробелоа.



Здесь U + 1 случайно совпало с числом пробелов, имевшихся в конце строки до начала переноса. Запишем алгоритм

```
            I := N − J;
МЕТКА 6:    I := I − 1;  S [I] := " "; IF I > N − J − (U+1)  THEN  GOTO  МЕТКА 6;
```

Затем следует Р уменьшить на единицу и, если Р еще не ноль, то вернуться к сдвигу следующего слова, добавив к J новое число пробелов:

```
    J := J + U;  P := P − 1;  IF P > 0  THEN  GOTO  МЕТКА 3;
```

Здесь можно заметить, что алгоритм готов и его надо лишь подряд весь записать:

```
            I := N + 1;
МЕТКА:      I := I − 1;  IF I = 0  THEN  GOTO  МЕТКА 2;
            IF S[I] = " "  THEN  GOTO  МЕТКА;
МЕТКА 2:    P := N − I;  IF P = 0  THEN  GOTO  OUT;
            J := −1;
МЕТКА 3:    J := J + 1;  S [N − J] := S [N − P − J];
            IF S [N − P − J − 1] /= " "  THEN  GOTO  МЕТКА 3;
            I := N − P − J − 1;
МЕТКА 4:    I := I − 1;  IF I = 0  THEN  GOTO  МЕТКА 5;
            IF S [I] = " "  THEN  GOTO  МЕТКА 4;
МЕТКА 5:    U := M − P − J − 1 − I;  I := N − J;
МЕТКА 6:    I := I − 1;  S [I] := " ";
            IF I > N − J − (U + 1)  THEN  GOTO  МЕТКА 6;
            J := J + U;  P := P − 1;  IF P > 0  THEN  GOTO  МЕТКА 3;
OUT:
```

В четвертой строке добавлен оператор выхода на метку OUT после того, как выясняется, что все пробелы из конца строки расставлены между словами (Р стало равно 0).

В качестае последнего примера разберем алгоритм распознааания заранее известных символов (стилизованных букв) на прямоугольном поле из точек и звездочек. Эта задача в равных вариантах а течение ряда лет предлагалась на конкурсах программистов, в том числе старших школьников в Красноярске и Томске. Анализ решения задачи позволит читателю, возможно, впервые, прикоснуться к проблеме технического зрения, оценить масштабы и сложность вопросов, возникающих у создателей "видящих" роботов и сканеров, оптическим способом читающих обычные тексты.

Итак, пусть на прямоугольном поле точек размерами M строк и N столбцов звездочками (★) изображены букаы П, Т, Г, О. Размеры букв произвольные, форма букв прямоугольная, толщина штриха − один символ. Минимальные ширина и высота − три символа. Ориентация букв − только прямая (исключаются наклоны и повороты). Букаы могут быть вложены одна а другую, но образующие их штрихи не могут соприкасаться. Заметим также, что а качестве симаолов фона и образующих могут быть взяты любые другие символы, например, пробел и знак "+". Количество разных букв неизвестно, его должна определить программа. В простейших вариантах этой задачи на поле вместо букв помещены вертикальные и горизонтальные отрезки. В других вариантах закладывают больше букв разных начертаний.

Пусть для анализа предлагается следующий даумерный символьный массив (прямоугольная матрица, или "картинка"). Если вы еще не разглядели на ней буквы, отодвиньте книгу примерно на метр от себя. Мы видим,

```
........................................
.*****.....****..********.******..****..
...*...***.*.....*......*.*....*..*.....
...*...*.*.*.***.*..***.*.*....*..*.....
.......***.*.*.*.*...*..*.*....*........
.*****.....*.*.*.*...*..*...............
.*...*.....*.*.*.*...*..*.***.***.***...
.*...*.....*.....*......*.*....*..*.*...
.*****...........********.*....*..***...
........................................
```

что всем условиям, постааленным в задаче, она удоалетворяет. Теперь подумаем, как определить прямоугольник в программе. Проще всего, если позволяет язык, задать двумерный литерный, или символьный, массив. В Паскале, Алголе-68, вариантах Бейсика это не составляет никакого труда. Опишем символьный массив S [1 : M, 1 : N], в котором первый индекс − номер строки, а второй − номер столбца. Каждый элемент S [I, J] − это один символ картинки.

Ввод в массив прямоугольника можно осуществить а даойном цикле. Во внутреннем цикле считываются элементы текущей строки, а во анешнем − осуществляется перебор строк. Значит, во внутреннем необходимо поменять значения индекса J от 1 до N с шагом 1, а во анешнем − индекса I от 1 до M также с шагом 1. После считыаания всех N символов текущей строки необходимо выполнить переход к новой строке. На Алголе-68 ввод будет выглядеть так:

```
  .FOR I  .FROM 1  .BY 1  .TO M  .DO
        .FOR J  .FROM 1  .BY 1  .TO N  .DO
             READ (S [I,J])
        .OD;  READ (NEWLINE)
  .OD;
```

Благодаря гибкости языка, запись можно сократить:

```
.FOR I .TO M .DO READ ((S[I, 1 : N], NEWLINE)) .OD;
```

Теперь добавим к "картинке" наши обозначения и подумаем, как распознать буквы.

```
         123                J ---->                     N  <-- номера
S[1,1]   .........................................   1 ]столбцов
         .******.....****..*********.*******..****..   2 ]
         ...*...****.*.....*.......*.*....*..*.....     ]
         ...*...*.*.*.****.*..****.*.*....*..*.....   I ]
         .......****.*.*.*.*...*..*.*....*.........   |  > номера
         .******.....*.*.*.*...*..*................   |  ] строк
         .*...*......*.*.*.*...*..*.****.****.****..  v ]
         .*...*......*.......*.......*.*....*..*.*...    ]
         .******...........**********.*....*..****...    ]
         .........................................   M ]
S[V,1]         S[I,J] - любой текущий символ         S[M,N]
```

На "картинке" видно, что все буквы отодвинуты от ве края, по крайней мере, на один символ. Это сделано специально для упрощения алгоритма распознавания. Присмотримся твперь к буквам, вернее, к их угловым точкам. Рассмотрим, например, лввый верхний угол и окружающие его соседние по ввртикали и горизонтали символы. Нетрудно увидеть, что вариантов всего два:

```
    .               .
1  .**  Т       2  .**  П.О.Г
    .               *
```

Для правого верхнего и левого нижнего углов тожв имевтся по два варианта:

```
    .             .             *            *
3  **.  Т.Г   4  **.  П.О   5  .**  О   6  .*.  Г.Т.П
    .             *             .            .
```

Как будет видно из дальнейшего изложения, анализа этих трех углов достаточно, чтобы однозначно распознать все букаы. В соответствии с принятыми обозначениями элемвнты любого угла определяется так:

```
          S[I - 1, J]   S[I, J]   центральный символ угла
                          /
S[I, J - 1]   .:.   S[I, J + 1]
                .
          S[I + 1, J]
```

Рассмотрим "картинку" слева направо и сверху вниз – так удобнее по нашим обозначениям. Сначала найдем левый верхний угол буквы. Он характерен тем, что при этом одновременно выполняются три равенства: S [I, J] = "*", S [I, J – 1] = ".", S [I – 1, J] = "." (см. варианты 1 и 2). Значит, перебирая символы "картинки" в поиске центрального элемента (звездочки) левого верхнего угла, нужно искать случай, чтобы слева и сввpху были точки. Заметим, что если к тому же точка окажется еще и снизу, то станет ясно, что обнаружена буква Т (вариант 1). В этом случае других углов буквы уже искать не нужно. В противном случае (вариант 2) обнаружена одна из букв П, Г или О и надо продолжать анализ. Если от левого верхнего перейдем к правому верхнему углу, то букву Г сможем выделить по наличию точки снизу (вариант 3), а неопределенность – П или О – останется. Чтобы отличить П от О, нужно от левого верхнего угла спуститься в левый нижний. Здесь П отличается (вариант 6) точкой справа.

Для подсчета количества каждой буквы введем целочисленные переменные P, G, T, O и обнулим их. Снова посмотрим на основную "картинку" и определим, в каком интервале должны меняться индексы I и J, если S [I, J] будем считать только центральным элементом левого верхнего угла. Ясно, что первым значением для I и J будет двойка (поскольку буквы краев не касаются). Чтобы определить последние значения, изобразим предельно маленькую букву О прижатой в правый нижний угол:

```
        S[M-3,N-3].....
          M-3 --> .***.   |
                  .*.*.   | I
                  .***.   v
                 ......  <-- M
      N-3
             J --> N
```

Очевидно, что левый верхний угол буквы в этом случае имеет координаты [M – 3, N – 3]. Следовательно, I = 2, 3, ..., M – 3; J = 2, 3, ..., N – 3. Заметим, что верхний (после левого) угол мы сможем обнаружить, если будем двигаться по строке до тех пор, пока вместо звездочки не встретим точку. Левый нижний угол найдем "симметричным" образом. Изобразим эти ситуации с дополнительными обозначениями:

```
          S[I,J]          S[I,J+U]
              \          /
              .*******.  <-- S[I,J+U+1]
              .*      *.
              .**      \
              /.\       S[I+1,J+U]
   S[I+W,J]     \ S[I+W,J+1]
             S[I+W+1,J]
```

Теперь сообразим, какие нужны циклы. Во-первых, внешний по строкам (I = 2,3,..., M – 3), во-вторых, внутренний вдоль строки (J = 2,3,..., N – 3), в-третьих, внутри внутреннего придется поместить еще два независимых цикла, в которых будет осуществляться поиск правого верхнего и левого нижнего углов. После распознавания буквы сразу выйдем во внешний цикл, давая такое приращение индексу J, чтобы выйти за пределы горизонтального штриха распознанной буквы (не искать левый верхний угол следующей буквы на образующем штрихе уже узнанной).

Циклы организуем условным переходом по метке. Текст программы запишем на Алголе-68, стараясь не употреблять никаких особенностей

языка. Комментарий отделим от основного текста сплошной чертой. Заметим также, что служебные слоеа в Алголе-68 обозначаются точкой впереди. Конец конструкции IF –THEN –ELSE отмечается слоеом .FI, а тела цикла – .OD; внутри этих конструкций скобки BEGIN и END необязательны.

```
.BEGIN .INT M,N;  M:=10;  N:=40;                  | задаем размеры картинки;
 .BEGIN  [1:M,1:N].CHAR S;                        | заводим основной массив;
  .FOR I.TO M.DO READ((S[I,1:N],                  | вводим картинку в
      NEWLINE)).OD;                               | массив S;
  .FOR I.TO M.DO PRINT((S[I,1:N],                 | распечатываем для
      NEWLINE)).OD;                               | контроля;
  .INT I,J,U,W,P,G,T,O;                           | заводим и
   P:=G:=T:=O:=0;  I:=1;                          | обнуляем счетчики;
M1:  I:=I+1;.IF I>M-3.THEN.GOTO OUT.FI;           | начинаем перебор строки;
     J:=1;                                        | начальный номер столбца;
M2:  J:=J+1;.IF J>N-3.THEN.GOTO M1.FI;            | перебор столбцов;
  .IF(S[I,J]="*".AND S[I-1,J]=".")                | ищем левый верхний угол
      AND S[I,J-1]="."                            | буквы;
  .THEN.IF S[I+1,J]="."                           | определяем, что это
        .THEN T:=T+1;  J:=J+1;.GOTO M2            | буква Т,
        .FI;  U:=1;                               | в противном случае
M3: U:=U+1;                                       | идем в правый
   .IF S[I,J+U]="*".AND                           | верхний
      S[I,J+U+1]="."                              | угол;
   .THEN.IF S[I+1,J+U]="."                        | выявляем
         .THEN G:=G+1;  J:=J+U;.GOTO M2           | букву Г,
         .FI;  W:=1;                              | в противном случае
M4:      W:=W+1;                                  | идем в нижний левый
       .IF S[I+W,J]="*".AND                       | угол;
           S[I+W+1,J]="."                         | нашли его;
       .THEN.IF S[I+W,J+1]="."                    | определяем
             .THEN P:=P+1;                        | букву П или
             .ELSE O:=O+1.FI                      | букву О;
              J:=J+1;.GOTO M2                     | уход на следующую букву;
       .FI;.IF I+W<M-1.THEN.GOTO M4.FI            | конец цикла по W;
   .FI;.IF J+U<N-1.THEN.GOTO M3.FI                | конец цикла по U ;
  .FI;  .GOTO M2;                                 | конец цикла по J ;
 OUT:  PRINT                                      | печатаем
     (("П-",P,"Г -",G," О-",O," Т-",T))           | результат;
 .END                                             | конец программы
```


После прогона данной программы на машине ЕС 1055М был получен результат:

П–2 Г–3 О–4 Т–3

Для отладки такой программы необходимо промежуточными печатями проследить за 1) нормальными изменениями индексов I и J; 2) индексами U и W.

Получие распечатку, по которой видно, как менялись индексы (и, следовательно, понимая, е какой последовательности и какие операторы исполнялись), необходимо сопостаеить введенную "картинку" с динамикой индексов, т.е. проследить, как выполнялась программа, и сраенить с тем, как она должна была выполняться. Как правило, после такого сопоставления ошибка а программе обнаруживается самим автором. (Более подробно об отладке см. раздел "Перевод программ на машинный язык. Отладка и тестирование").

Приведвнный алгоритм заметно усложнится, если мы допустим наличие дефектов на изображении, например, пропадание одной звездочки в любом месте буквы, но так, чтобы зрительно букву можно было еще отличить от других, или появление лишней звездочки, "прилипшей" к образующему штриху. Сложности такого же рода возникнут и в случае анализа более реальных шрифтов (начертаний) букв, чем использоеанный прямоугольный.

Итак, читатель теперь должен проникнуться убеждением, что точные аккуратные рисунки, раскрывающие все существенныв детали алгоритма, – незаменимое средство и надежная опора в трудном переходе от задачи к программе. Многолетний опыт общения автора с начинающими программистами свидетельствует, что небрежное отношение к наглядному изображению создаваемого алгоритма есегда кончается одним и тем же – обилием ошибок в программе и большими затруднениями при ее отладке.

Остановимся на чисто внешней стороне записи текста программы. Речь идет о том, как следует размещать его. Транслятору и машине есе равно, будет ли текст на Паскале или Алголе четко структуриреан с визуальным выделением меток, вложенных циклое, блокое и т.д., или же он полностью заполнит все пространство каждой строки. Красивый внешний вид программы нужен прежде всего ее автору. Аккуратная запись программы позволяет ее легко читать и понимать, поэтому нужно во всех случаях стремиться к хорошему оформлению результатов работы программиста. К сожалению, этим правилом многие пренебрегают, особенно когда пишут программы "на скорую руку", для себя. При создании программного продукта на продажу (что, быть может, в наших услоеиях еще непривычно звучит) требование эстетического оформления становится неизбежным. Это дизайн в программировании.

В заключение отметим, что последние две задачи могут служить тестом на наличие одной из обязательных черт профессионального программиста – способности мысленно в правильной последовательности

3*

представить и со скрупулезной точностью записать в мельчайших деталях весь алгоритм решения. Постоянно проявлять изобретательность и терпение на уровне, по крайней мере, таких задач – удел программиста.

Что касается алгоритма распознавания печатных символов оптическим сканером, то мы разобрали лишь самый простой, частный случай. При добавлении новых символов или изменении начертания уже заложенных алгоритм требует существенной доработки.

Более простой вариант алгоритма, не требующий изменений при смене шрифта и даже алфавита, состоит в следующем. Подбирается матрица из большого числа мелких точек (например, 30х40) для того, чтобы с ее помощью можно было однозначно предстаaить любые буквы и символы, применяемые в книгопечатании, размером, например, 4х3 мм. Каждый используемый символ изображается точками на матрице, кодируется битами и заносится в память машины. Оптическая система сканера строит изображение читаемого текста также в виде набора светлых и темных точек, которые передаются в память. Всевозможные дефекты на бумаге (грязь, недопечатка отдельных участков) отобразятся лишними и недостающими единицами на фоне нулей.

Процесс последующего распознавания заключается в том, что сначала выявляются границы буквы (например, по левому верхнему углу или другому признаку), а затем берется прямоугольник точек-бит и сравнивается побитно на полное совпадение с хранящимися в матрицах образами. Поскольку на печатном тексте всегда есть хотя бы незначительные дефекты, точное совпадение всех бит двух образов одного символа (хранимого и прочитанного) будет достигаться далеко не всегда. Часто будут наблюдаться несовпадения в отдельных точках. Если этих несовпадений мало, то принимается решение о том, что образы идентичны и, значит, распознан определенный символ.

Читателю уже должно быть ясно, что скорость распознавания у такого общего алгоритма в принципе меньше, чем предыдущего, более частного, так как приходится сравнивать прочитанный символ в среднем с половиной хранимых. Поэтому, чтобы сканер мог быстро читать тексты, нужны быстрый процессор и оптимизированный алгоритм. На качестве распознавания, т.е. количестве допускаемых ошибок на единицу объема текста, сказываются освещенность и контраст оригинала, его полиграфическое качество. Впрочем, для исключения влияния внешнего освещения многие сканеры снабжаются собственными, особой конструкции источниками света.

Метод пошагового уточнения

Из рассмотренных выше примеров построения алгоритмов решения сравнительно простых задач легко заметить, что программирование – это сложное и трудоемкое дело и важны любые усовершенствования, способствующие подъему производительности труда программистов. Специалисты давно уже размышляют и спорят о том, какой должна быть

технология написания программ. Один из приемов, доказавших свою эффективность, – это метод пошагового или последовательного уточнения, названный методом проектирования "сверху вниз", или нисходящим методом. Как нам представляется, между пошаговым (последовательным) уточнением и нисходящим проектированием есть небольшое отличие, которое состоит в том, что термин "пошаговое уточнение" употребляют в основном применительно к процессу создания относительно небольших программ, а нисходящим проектированием чаще называют один из способов разработки крупных программных комплексов.

Суть метода пошагового уточнения заключается в том, что разработку алгоритма от постановки задачи к законченному программному тексту ведут от общего, поначалу неясного наброска к постепенно детализируемой, прорабатываемой и уточняемой в отдельных элементах картине. Начинают разработку на естественном языке, записывая набором последовательных предложений те действия, которые должна выполнять машина для решения поставленной задачи. Затем раскрывают смысл отдельных предложений более подробной записью. По мере того, как текст на естественном языке по точности выражения машинных операций и внешнему виду приближается к конструкциям языка программирования, вместо обычных предложений выписывают готовые операторы. Таким образом, в процессе разработки создается несколько экземпляров программы разной степени детализации – от общего наброска на естественном языке до окончательного текста на алгоритмическом языке.

Опытные программисты первые наброски "прокручивают" в уме, а на бумаге или экране дисплея работу ведут "с середины", записывая сразу довольно подробный текст со многими заготовками уже на языке программирования. Начинающим рекомендуется проявлять терпение и записывать все стадии порождаемой программы. Каждый очередной вариант должен быть подвергнут критическому анализу, прежде чем последует его уточнение.

Процесс создания больших программ состоит в следующем. Пусть необходимо разработать программу P для решения задачи T. Если T – сложная задача, то ее разбивают, например, на подзадачи T1, T2, T3, реализуемые подпрограммами P1, P2, P3, каждая из которых проще и меньше P. В свою очередь, подзадача T1 разбивается на подзадачи T11, T12, T13, T14, реализуемые подпрограммами P11, P12, P13, P14. Аналогично поступают и с другими сложными подзадачами. Таким образом, получается иерархическая структура, изображенная на рис.28.

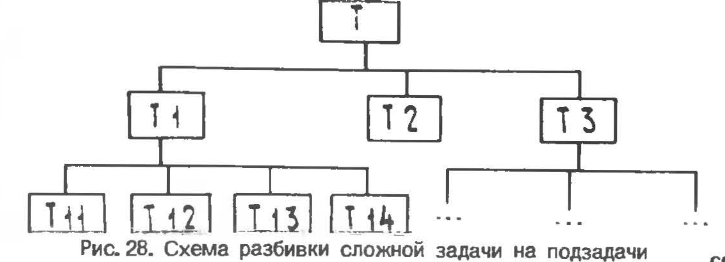


Рис. 28. Схема разбивки сложной задачи на подзадачи

Идея нисходящего проектирования в этом случае состоит в том, что сначала строится программа-организатор верхнего уровня Р, а которой для решения подзадач вызываются подпрограммы следующего уровня. Сами тексты подпрограмм пишутся после того, как будет готова программа верхнего уровня. Если подпрограммы, в свою очередь, вызывают еще какие-то подпрограммы следующего уровня, то их тексты пишутся после вызывающих подпрограмм. Получающаяся таким образом структура подпрограмм повторяет структуру подзадач и этапность работы над ними.

При восходящем (снизу вверх) проектировании сначала разрабатываются подпрограммы нижнего уровня, затем следующий слой выше и так вплоть до головной, ведущей программы, в которой вызываются все подчиненные подпрограммы.

И в том, и в другом методе возникает проблема отладки. Когда готова головная программа, но нет еще текстов подпрограмм, то полную отладку головной провести невозможно. Вместо отсутствующих подпрограмм приходится писать простейшие программы-имитаторы ("заглушки"). Если идти снизу вверх, то для проверки работоспособности готовых подпрограмм необходимы вызывающие программы (драйверы), обеспечивающие прогон подпрограмм во всех режимах их будущей работы с широким набором значений входных параметров. В обоих случаях приходится выполнять дополнительную работу – писать "заглушки" или драйверы.

Сравнение языковых конструкций

Приведем основные конструкции наиболее распространенных языков программирования Фортран-4, Фортран-77, Алгол-60 (в варианте Алгол-ГДР), Алгол-68 (в варианте ЛГУ), Паскаль, Ада. Для краткости будем обозначать языки соответственно: Ф4, Ф77, А60, А68, П, АДА. Будем давать не все допустимые, а только базовые варианты конструкций, поскольку в некоторых языках существует много разновидностей одной и той же конструкции. Отметим также, что указанное в конструкции малыми русскими буквами содержимое терминологически не везде строго соответствует тому, что написано в руководствах по конкретным языкам. Например, в описании Фортрана-77 можно встретить термин "инструкция", который в данном тексте заменен на термин "оператор". Замены произведены для простоты и единства изложения.

Цель данного пункта – показать, что языки "вычислительной группы", особенно так называемые алголоподобные, не значительно отличаются друг от друга. Кроме того, такое сравнение позволяет легче усвоить суть той или иной конструкции и меньше обращать внимания на ее форму. Хотя, разумеется, никакой транслятор "не простит" вам вольности, если вы, например, в программе на Фортране запишете цикл по правилам Паскаля. Трансляторы-полиглоты еще не появились.

В этот обзор не вошел Бейсик, поскольку по нему выпущено уже очень много литературы. Любителям Бейсика можно порекомендовать

ознакомиться с новой версией, так называемым быстрым Бейсиком (QBASIC), появившимся недавно в рамках MS DOS [5]. Значительная модернизация языка с отходом от обязательного примитивизма, включение в него многих новых черт и возможностей ставят QBASIC в один ряд с такими развитыми языками, как Ада, Алгол-68, Фортран-77. А что касается использования графики и звука, то QBASIC их даже превосходит. Многочисленные достоинства QBASIC, несомненно, привлекут к нему внимание пользователей ПЭВМ.

В обзоре также отсутствует достаточно популярный на малых и персональных ЭВМ язык Си. Объяснение этому факту банальное – к моменту написания данного раздела автору не довелось познакомиться с этим языком сколько-нибудь полно. В оправдание заметим, что язык Си в основных своих чертах близок к языкам Алгол-68 и Ада, а некоторые конструкции отличаются совершенно незначительно.

1. *Имя* (идентификатор) переменной или массива во всех языках выглядит практически одинаково, т.е. как произвольная последовательность букв и цифр, начинающаяся с буквы. Максимальная длина имени может быть ограничена (например, не более 6 символов). В отечественных и адаптированных зарубежных трансляторах допускается употребление в именах чисто русских букв, не имеющих аналогов в латинском алфавите (таких, как Ш, Щ). Почти во всех трансляторах, в том числе отечественных, служебные слова записываются на английском языке. Поскольку их общее число в языке не превышает нескольких десятков, проблем понимания и перевода обычно не возникает.

2. *Простые переменные.* Во всех языках имеются три типа простых переменных: целые (обозначаются служебным словом INTEGER или INT), вещественные или реальные (REAL, FLOAT), логические, или булевские (LOGICAL, BOOLEAN, BOOL). Для того, чтобы машина могла правильно работать с переменными названных типов, они должны быть в программе описаны, т.е. снабжены указателями типов. Сведения о типах приводятся в списках переменных в начале программы. В языках Ф4 и Ф77 допускается не описывать целые и вещественные переменные. В таком случае целыми считаются те, имена которых начинаются буквами I, J, K, L, M, N, а все остальные – вещественными.

Ф4, Ф77	REAL A, B, I1, MAL INTEGER C1, C2, X LOGICAL F, L
А60	'REAL' A, B, I1, MAL; 'INTEGER' C1, C2, X; 'BOOLEAN' F, L;
А68	.REAL A, B, I1, MAL; .INT C1, C2, X; BOOL F, L;
П	A, B, I1, MAL : REAL; C1, C2, X : INTEGER; F, L : BOOLEAN;

АДА A, B, I1, MAL : real; C1, C2, X : integer;
F, L : boolean;

С целью улучшения использования оперативной памяти в большинстве языков предусмотрена возможность раздельного описания коротких и длинных целых переменных (имеющих значительно отличающиеся диапазоны изменений), а также коротких и длинных вещественных, имеющих разную точность (т.е. с короткой и длинной мантиссой соответственно).

Изображения целых, вещественных и логических значений в текстах программ должны удовлетворять следующим несложным правилам. Значения целых не должны выходить за пределы своего диапазона, который можно найти в руководстве по выбранному языку для конкретного семейства ЭВМ. Знак плюс у положительных чисел необязателен. У вещественных дробная часть должна отделяться от целой части десятичной точкой (не запятой!). Количество значащих цифр не должно быть больше, чем допускается в мантиссе для данного типа переменных. Например, на ЕС ЭВМ вещественные обычной длины (одинарной точности) в Фортране и Алголе-68 имеют 7 гарантированных десятичных цифр в мантиссе. Диапазон представления вещественных также можно найти в руководстве по языку. На наиболее распространенных в нашей стране машинах он составляет не менее $10^{\pm 38}$. Для изображения очень больших и очень маленьких чисел рекомендуется применять нормализованный формат записи вещественных.

Логические значения TRUE и FALSE записываются в апострофах (А60), с точкой впереди (А68), в обрамлении точек (Ф4, Ф77) или как указано (П, АДА). Точно так же должны быть представлены значения переменных при вводе их в ходе исполнения программы. При выводе переменные названных типов оформляются аналогично, но уже машиной.

3. *Арифметические выражения* записываются с употреблением переменных и элементов массивов, знаков четырех арифметических действий (+ – */, в качестве знака умножения x применяется звездочка), возведения в степень (**), арифметических (SQRT – корень квадратный, ABS – абсолютная величина и т.п.), тригонометрических (SIN, COS и т.п.) и других функций. Полный список встроенных функций можно найти в руководстве по языку. Для регулирования порядка выполнения действий применяются круглые скобки. Правила записи арифметических выражений в разных языках отличаются незначительно.

4. *Логические выражения* (условия) строятся из целых, вещественных и логических переменных с употреблением знаков отношения (=, <, >, >=, <=, <> или /=) и логических операций (AND, OR, NOT). В Ф4 и Ф77 в качестве знаков отношений записываются аббревиатуры соответствующих английских наименований: .GT. (greater than) вместо >, .LE. (less or egual) вместо <= и т. д.

5. *Оператор присваивания* во всех языках имеет один и тот же вид:

имя < служебный символ > выражение

Небольшая разница состоит в служебном символе, в качестве которого употребляют знаки "=" в Ф4, Ф77 и ":=" ("двоеточие" и "равно") во всех остальных. "Имя" означает наименование (индентификатор) переменной в программе; "выражение" – любое арифметическое или логическое выражение вроде A + 25.3 * B – C/X, содержащее хотя бы одно число или переменную. Символ "пробел" можно писать всюду, за исключением имен, служебных слов, служебных символов и чисел. Например, A = X**3 – 2.718   X := (X + A)/3.14

6. *Оператор ветвления* (IF – THEN – ELSE = если–то–иначе). Смысл его состоит в том, что машина сначала проверяет выполнение условия, стоящего после IF, а затем исполняет то, что находится после слова THEN до ELSE (если условие истинно), или то, что стоит после ELSE (если условие не выполняется).

Ф4 IF (условие) K, L, M (K,L,M – целочисленные метки для переходов при разных значениях условия)
IF (условие) THEN оператор
Ф77 IF (условие) K, L, M
IF условие THEN операторы ELSE операторы END IF
П IF условие THEN оператор или блок
ELSE оператор или блок
АДА if условие then операторы else операторы end if

Во всех языках часть оператора IF – THEN – ELSE, начиная со слова ELSE, может отсутствовать. После ELSE всегда можно писать новый IF, а после THEN – не во всех языках.

7. *Оператор цикла* состоит из заголовка и тела. В заголовке определяется переменная, называемая параметром цикла, и задаются интервал и шаг ее изменений. Существуют и другие варианты заголовков цикла. Тело цикла состоит из группы (или всего одного) операторов, которые выполняются несколько раз, причем каждый раз с новым значением параметра. Всюду в следующих ниже записях цикла I – переменная-параметр цикла, N1, N2, N3 – выражения (или просто имена, или конкретные числа), задающие соответственно первое, последнее значения параметра цикла и шаг его изменения между этими крайними значениями. Указанные в квадратных скобках части могут отсутствовать, сами квадратные скобки в этих местах никогда не пишутся в программе. Справа от конструкции дано пояснение.

Ф4, Ф77	DO M I = N1, N2 [, N3]	заголовок цикла
	операторы	тело
	M последний оператор	цикла

Примечание. *В качестве M указывается целое число – метка, помечающая последний оператор, входящий в цикл; им может быть специальное служебное слово CONTINUE.*

А60	'FOR I:=N1 'STEP' N3 'UNTIL' N2 'DO' оператор или составной оператор	заголовок цикла тело цикла
А68	[[.FOR I[.FROM N1][.BY N3]].TO N2].DO операторы .OD	заголовок тело

Примечание. *Если N1 или N3 равно 1, то его можно не указывать вместе со своим служебным словом. Если параметр цикла в его теле не используется, то весь заголовок можно сократить до .TO N2.DO, чтобы задать точное число повторений тела цикла. А если и оно не известно, а определяется, например, внутри цикла по ходу вычислений, тогда цикл начинают сразу со слова .DO.*

П	FOR I:=N1 TO N2 DO оператор или составной оператор	заголовок цикла тело цикла

Примечание. *Если* N2 < N1, *то необходимо писать* DOWNTO *вместо* TO.

АДА	for I in [reverse] дискрет. диапазон loop операторы end loop	заголовок тело

8. *Оператор безусловного перехода* заставляет процессор ЭВМ перейти к выполнению оператора, находящегося в другом месте программы и помеченного меткой-приемником. При отсутствии операторов перехода программа выполняется машиной последовательно сверху вниз, оператор за оператором.

		пример
Ф4, Ф77	GOTO метка-источник	GOTO 4
А60	'GOTO' метка-источник	'GOTO' MET1
А68	[.GOTO] метка-источник	[.GOTO] MET1
П	GOTO метка-источник	GOTO 5
АДА	goto метка-источник	goto MET2

Примечание. *В А68 слово .GOTO необязательно.*

9. *Метка-источник* – это указанная в операторе GOTO точка программы, на которую надо сделать переход. Метка-источник стоит в том месте, откуда делается переход. Метка-приемник – имя (адрес) той точки (оператора) в программе, куда, возможно, будет сделан переход при исполнении программы. Любая метка-приемник всегда присутствует в программе (блоке или подпрограмме) только один раз. Каждый оператор, на который требуется выполнить переход, помечается уникальной меткой-приемником. Метка-источник в операторах перехода может упоминаться любое количество раз, в том числе и ни одного.

Ф4, Ф77	целое без знака; метка-приемник всегда располагается в 1...5 позициях строки, т.е. слева от основного текста программы
А60, А68	имя; метка-приемник отличается наличием справа двоеточия, например, MET76:
П	целое без знака; метка-приемник имеет справа двоеточие, например, 53:
АДА	имя; метка-приемник заключается в пары знаков "меньше" и "больше": <<MET1>>

10. *Комментарий* – пояснения, которые автор программы заносит в нее для себя или для других людей (не для машины!). Содержимое комментариев никак не влияет на исполнение программы. В комментарий можно включать любые символы, кроме тех, которые являются его ограничителями.

Ф4, Ф77	C (в первой позиции строки) – текст пояснения до конца строки
А60	'COMMENT' текст пояснения;
А68	# текст пояснения #
П	(* текст посянения *)
АДА	-- текст пояснения (до конца строки)

11. *Разделитель операторов*

Ф4, Ф77	на одной строке нельзя писать больше одного оператора, поэтому разделитель не нужен

А 60, П, АДА каждый оператор заканчивается точкой с запятой

А68	операторы отделяются друг от друга точкой с запятой. Перед закрывающей скобкой любого типа (.FI, .OD, .END, ")" и т.п.) точка с запятой не ставится.

12. *Составной оператор и блок.* Блочная структура существует во всех "алголоподобных" языках. Она строится из составных операторов и собственно блоков. Составной оператор – это группа операторов, заключенная в так называемые операторные скобки BEGIN и END. Он может находиться на тех же местах, что и обычные операторы. Составной оператор, помещенный внутри условного после THEN или ELSE, исполняется или не исполняется (пропускается) целиком. В этом главный смысл введения конструкции "составной оператор". В языках А68, АДА, благодаря наличию закрывающих скобок у многих конструкций, таких как .FI, .OD, end if, end loop и т.п., операторные скобки в большинстве случаев не требуются. В А68 в остальных случаях BEGIN и END можно заменять на (и).

Если внутрь составного оператора поместить описания используемых в нем переменных, то получится, строго говоря, блок. Внутри блока локализуются все описанные в нем переменные, т.е. за пределами блока ни до входа в него, ни после выхода из него эти переменные не существуют, и их никак нельзя использовать.

13. *Встроенные (стандартные) функции* имеются практически во всех языках. Это такие общеупотребительные математические функции, как SQRT (квадратный корень), SIN (синус), LOG (логарифм), SIGN (знак числа) и др. Число функций в языках различно, их список и правила пользования можно найти в руководстве по языку.

Каждый транслятор "знает" имена встроенных функций своего языка, поэтому их не надо никак описывать в программе. В библиотеке транслятора для любой функции имеется своя подпрограмма в машинных кодах (или одна подпрограмма на группу близких функций, например, тригонометрических). Копия текста нужной подпрограммы включается в состав рабочей программы при ее сборке. Наиболее простые функции (SQRT, SIGN и аналогичные) не имеют специальных подпрограмм, а реализуются одной или несколькими машинными командами, которые "знает" транслятор.

14. *Начало и конец* программы

Ф4, Ф77	PROGRAM ИМЯ текст программы STOP END	обязательно не для всех трансляторов
А60	'BEGIN' текст программы 'END"	
А68	.BEGIN текст программы .END или (...)	

Примечание. *В большинстве языков выполнение программы останавлиеается (прекращается) не только выходом за ее конец, но и применением специального* оператора STOP.

П	PROGRAM ИМЯ; описания BEGIN операторы END.
АДА	выделенного слова нет текст программы end [имя]

15. *Массивы* (переменные с индексами) имеются во всех языках. В качестве индексных скобок в Ф4, Ф77, АДА используются символы ( ), а в А60, А68, П – [ ]. Квадратные скобки, несомненно, здесь удобнее. Сохранение круглых для этих целей в таких сравнительно новых языках, как Ф77 и АДА, – определенная дань традиции и желание использовать накопленные программы и технические средства (клавиатуры, перфораторы и др.), в которых квадратные скобки отсутствуют.

Описания массивов обязательны во всех языках: транслятор должен "знать" его размер, количество измерений (индексов), переделы изменений индексов, тип элементов. Вся эта информация дается в описании. Тип всех элементов одного массива должен быть один и тот же, в этом основное отличие массивов от более сложных объектов программы – структур, записей, файлов. Рассмотрим в качестве примера два массива – одномерный из вещественных и двумерный из целых элементов. Одномерный с числом элементое 10, а двумерный – в виде таблицы из трех строк и пяти столбцов. Нижнюю границу по каждому измерению будем считать равной единице. Определенные таким образом массивы с постоянными в ходе исполнения программы границами называются **статическими.**

Ф4, Ф77	REAL A(10) INTEGER B(5,3)
А60	'ARRAY' A [1:10]; 'INTEGER' 'ARRAY' B [1 : 3,1 : 5];
А68	[10] .REAL A; [3,5] .INT B;
П	A: ARRAY [1..10] OF REAL; B: ARRAY [1..3,1..5] OF INTEGER;
АДА	A: array (1..10) of REAL; B: array (1..3,1..5) of INTEGER;

Динамические массивы. т.е. массивы с изменяемыми при выполнении программы размерами, возможны в А60, А68, АДА. В их описаниях в качестве выражений для верхних и нижних границ индексов можно указывать имена и арифметические выражения, значения которых вводятся или вычисляются при исполнении программы.

16. *Операции ввода-вывода* осуществляют пересылку в оперативную память с переводом во внутреннее представление исходных данных с клавиатуры, магнитных дисков, лент и других внешних устройств и выдачу из оперативной памяти с обратным преобразованием на бумагу, экран дисплея, диск, ленту и т.д. В последние годы быстро меняются характеристики внешних устройств и наши представления о том, какой вид должны иметь результаты исполнения программы. Еще 20 лет назад распечатки результатов представляли собой, как правило, сплошные колонки чисел, иногда перемежающиеся словесными пояснениями. Сегодня формирование выдачи для наилучшего восприятия человеком выделилось в особый раздел программирования, часто и серьезно рассматриваемый на конференциях по информатике, в том числе международных.

Проблема состоит в том, что все распространенные ныне языки программирования сложились минимум десятилетие назад и встроенные в то время в них конструкции ввода-вывода плохо сочетаются с современными техническими средствами и возможностями. Поэтому разрабатываются многочисленные расширения языков, библиотеки подпрограмм ввода-вывода, графические пакеты и др. Чтобы не потеряться в массе деталей и отличий, рассмотрим здесь только самы простые варианты стандартных операторов ввода-вывода, а за более подробными разъяснениями отошлем к другим разделам и руководствам по конкретным языкам.

Наиболее просто операторы ввода-вывода выглядят в алголоподобных языках:

А60	READ (ИМЯ 1, ИМЯ 2,..., ИМЯ N); PRINT (ИМЯ 1, NEWLINE, ИМЯ 2,..., ИМЯ N);
А68	READ ((ИМЯ 1, ИМЯ 2,..., ИМЯ N)); PRINT ((ИМЯ 1, NEWLINE, ИМЯ 2,..., ИМЯ N)); WRITE ((ИМЯ 1, NEWLINE, ИМЯ 2,..., ИМЯ N));

П READ (ИМЯ 1, ИМЯ 2,..., ИМЯ N);
WRITE (ИМЯ 1, ИМЯ 2,..., ИМЯ N);
WRITELN (ИМЯ 1, ИМЯ 2,..., ИМЯ N);

Примечание. *Во всех примерах первый оператор (READ) обеспечивает ввод символоа из внешней среды, преобразование, если нужно, во внутреннее представление (например, вещественных значений) и размещение в тех ячейках памяти, которые назначены транслятором для переменных ИМЯ 1, ИМЯ 2 и т.д. Второй (и третий, где он показан) оператор переводит содержимое ячеек памяти, обозначенных именами, во внешнее (символьное) представление и выдает на периферийное устройство. На самом деле механизм ввода-вывода устроен сложнее, но приводить его подробно здесь необязательно.*

Ф4, Ф77 READ*, ИМЯ 1, ИМЯ 2,..., ИМЯ N
PRINT*, ИМЯ 1, ИМЯ 2,..., ИМЯ N

АДА GET (ИМЯ 1); GET (ИМЯ 2); ... GET (ИМЯ N);
PUT (ИМЯ 1); PUT (NEWLINE); PUT (ИМЯ 2); ... PUT (ИМЯ N);

В каждой операционной системе этим стандартным функциям предписаны определенные типы устройств. Во многих случаях это клавиатура для ввода и экран для вывода. В ПЭВМ названное соответствие является общепринятым. Для ввода символы (например, цифры одного числа, представляющие собой изображение значения) подаются один за другим. Соседние значения разделяются пробвлами. При выводе значения каждого типа представляются в некотором стандартном для данного транслятора виде. В большинстве случаев стандартный вывод пользователя не устраивает, потому что он нвдостаточно рационален (много лишних цифр, неэкономно расходуется место на бумаге или экране), тогда в Фортране применяют формвты, в Алголе – специальные процедуры, в Паскале – параметры после имвн переменных.

Очень важную роль в языках играет модульность, т.е. возможность выделения частей алгоритма в самостоятельные программные единицы – модули. Первые модульные конструкции – функции и подпрограммы – появились давно, практически вместе с первыми языками высокого уровня – Фортраном, Алголом-60. В более поздних языках принцип модульности очень сильно развит и обогащен новыми идеями. Подробный рассказ о модульности занял бы слишком много места, поэтому ограничимся лишь самыми простыми вариантами модульных конструкций.

Основное назначение модуля – сосредоточить внимание программиста-разработчика на создании алгоритма, пригодного для применения в рвзных программах. Копии одного и того же модуля, например, для расчета широко известной математической функции, могут применяться параллельно и незаaисимо во множестве программ. Тем самым экономится труд программиста. Обширная библиотека аккуратно составленных и тщательно отлаженных модулей – одно из главных богатств профессионального программиста. Библиотека модулей незвменима при создании большой программы методом "снизу вверх", напоминающим строительство здания из отдельных готовых блоков.

Модульность позволяет не обращать внимания на состав переменных в вызывающей программе, тем самым обеспечивая простоту распределения работ над крупным проектом между многими исполнителями. Принцип модульности вообще характерен для производства сложных технических изделий, в которых всегда выделяются отдельные блоки и узлы. Вспомним хотя бы в общих чертах устройство телевизора, автомобиля или самолета. В наших языках модулями являются функции, процедуры, подпрограммы.

Модуль-функция по своему назначению и использованию ничем не отличается от стандартной встроенной функции, например синуса. Функция может иметь любое число аргументов (параметров), в том числе и ни одного. Пример функции без параметров – процедура TIME, которая в момент обращения к ней считывает с таймера машины текущее время суток и выдает его в программу в виде строки симаолов, например, 11 : 28 : 05.

Описание модуля состоит из заголовка и тела. Заголовок обязательно содержит специальное служебное слово и имя модуля. Если он имеет параметры, называемые в описании формальными, то в заголовке они должны быть специфицированы, т.е. указаны их тип и способ вызова. В качестве служебных применяются английские эквиваленты слов "подпрограмма", "процедура", "функция". Использование модуля состоит в обращении к нему по имени с перечислением фактических параметров. К однажды описанному модулю можно обращаться в программв (т.е. заставлять его отработать) многократно, в любой нужный момент.

Довольно тонким моментом является различие в способах вызова параметров модуля по имени и по значению. В рабочвй программе, напомним, все имена переменных заменены на адреса (номера ячеек памяти), в которых эти переменные размещаются. При вызове по имени внутрь модуля вместо формального параметра передается адрес переменной – фактического параметра. Внутри модуля значение переменной, т.е. содержимое соответствующей ячейки, может измениться и сохранится измененным после выхода из модуля. Вызов по имени используется для параметров, которые являются выходными ("выходят из модуля").

Для формальных параметров, вызываемых по значению, отводятся свои ячейки памяти, которые заполняются копиями значений фактических параметров. В таком варианте вызова значения переменных, употребленных в качестве фактических параметров, не могут изменяться никакими действиями внутри модуля. Твм самым переменные вызывающей программы защищаются от, возможно, неправильной работы модуля. Вызов по значению имеет смысл для входных параметров. Оба способа вызова параметров реализованы в А60, А68, П. В оформлении модулей довольно много отличий и тонкостей, подробное объяснение заняло бы слишком много места, поэтому вряд ли оно уместно в данной книге.

В наших языках имеются (в разных объемах) средства описания и манипулирования символьными и строковыми переменными и массивами. Простые строки пояснений, направляемых на выдачу вместе со значениями переменных, возможны во всех языках и должны быть заключены в апострофы (АДА, Ф4, Ф77, П), двойные апострофы (А60), двойные кавычки (А68). Такие строки можно писать в операторах вывода всюду, где допускается указывать имена переменных. За более подробной информацией по употреблению символьных и строковых переменных читателю придется обратиться к учебнику по выбранному языку программирования.

В заключение данного раздела отметим, что в нем показаны далеко не все конструкции и возможности таких богатых языков, как Ада, Алгол-68, Фортран-77.

РЕДАКТОРЫ И ПОДГОТОВКА ТЕКСТОВ

Важнейшим средством подготовки всевозможных текстов на ЭВМ является специальная программа – твкстовый редактор. Существует много редакторов – простых и сложных, униаерсальных и специализированных, предназначенных для определенных видов работ. Различные инструментальные программные средства имеют собственные, встроенные текстоаые редакторы. Мы же подробно рассмотрим основные функции редактора, схему его работы, команды и функциональные клавиши, дадим несколько полезных советов.

Осноеные функции редактора – это работа с файлами: показ текста (файла) на экране, вывод на бумагу; вставка строк и символов; удаление строк, слов и символов; перемещение нескольких строк или прямоугольного блока символов в другое место текста; копирование (размножение) строк и цепочек символов; поиск строки по заданному образцу (контексту, цепочке символов); выравнивание ширины текста, автоматический перенос слов по правилам грамматики языка; резка и склейка строк; замена одного абзаца (повторяющейся цепочки символов) на другой; преобразование строчных букв в прописные и обратно; вставка в текст заготовок из определенного набора, быстрая смена набора заготовок; контроль формируемого текста по правилам грамматики и словарю выбранного языка; набор сложных математических формул; полиграфическое оформление текста.

Из этого неполного перечня видно, что в принципе с помощью редактора можно выполнять много полезной работы над текстом. Задача состоит лишь а том, чтобы аыбрать наиболее подходящий, удобный редактор и научиться им пользоваться. Поясним отдельные функции.

Работа с файлами. Сформированный пользователем текст необходимо сохранить на диске, т.е. создать из него файл. Для продолжения работы любой ранее записанный текст должен считываться редактором с диска. Следует обеспечить копирование в текущий (изготовляемый) текст любого числа строк из любых имеющихся на диске файлов. Для подстраховки на случай сбоя машины или аварийного отключения электроэнергии редактируемый текст должен периодически записываться на диск по команде или автоматически.

Показ текста на экране. Для того, чтобы автор мог просматривать создааемый текст, находящийся во время работы в оперативной памяти компьютера, существует несколько возможностей. Простейшая состоит в выводе на дисплей по команде группы строк а заданном интервале их номеров. Только такая возможность и была в первых, самых простых редакторах. Более сложный, с точки зрения программного воплощения, еариант – организация на экране "окна", в котором постоянно виден фрагмент, объемом в 15...25 строк. Пользователю дается возможность "прокрутки" текста в окне вверх и вниз как плавно, так и полными окнами (экранами, страницами). Если длина строки текста больше ширина окна, то допускается перемещение изображения в окне влево-вправо. Обеспечивается быстрое перемещение с любого места на начало или конец файла. Некоторые редакторы позволяют создавать несколько окон и работать одновременно с несколькими файлами или частями одного файла.

Вывод (печать) *текста* на бумагу возможен, еспи в составе ЭВМ имеется принтер. Собственно печать не всегда входит в функции редактора, тогда она выполняется срвдствами ядра операционной системы. Поэтому подготовленный текст надо сначала записать на диск, выйти из редактора, а затем командой ОС вывести файл на печатающее устройство. В редакторах, предназначенных для формирования многошрифтовых многоалфавитных текстов, функция вывода на печать встроена внутрь.

Вставка строк и символов. Вставка символов является наиболее частым режимом работы, установленным по умолчанию либо нажатием специальной клавиши. В режиме вставки часть строки, расположенной правее курсора, отодаигается при нажатии символьных клавиш. При отмене этого режима ноеые симаолы замещают те, что были на их месте. Часто новички не догадываются о последней возможности и сначала затирают ошибочные символы клааишей пробела, затем возвращают курсор на начало чистого места и лишь после этого набирают правильный текст.

Вставка строк осуществляется путем раздвижки текста, освобождения места под новые строки и ввода их. Отодвигают строки нажатием одной из клавиш-стрелок или командой вставки строк. Иногда группа строк вводится как добавление послв твкущей строки. Тогда все последующие автоматически отодаигаются.

Перемещение группы подряд идущих строк на новое место возможно в большинстве редакторов и выполняется по команде, в которой надо указать или отметить первую и последнюю из перемещаемых строк и то место, куда их следует помвстить. Эта операция напоминает работу

с ножницами и клеем над текстом, напечатанным на одной стороне бумаги. Копирование строк отличается от перемещения тем, что оригинал остается не своем месте, а на новое заносится копия.

Удаление символов – простая операция, заключающаяся в перемещении части строки, находящейся правее курсора, влево. При этом символ, расположенный над курсором, исчезает. Для удаления символов обычно используется специальная клавише.

Удаление строк – аналогичная операция, но она более опасна, поскольку большинство редакторов не могут восстановить ошибочно удаленные строки и их придется набирать заново. Операция выполняется либо специальной клавишей-стрелкой, либо командой с параметрами (диапазоном номеров удаляемых строк). После удаления строк необходимо внимательно оценить получившийся результат, прежде чем двигаться дальше.

После вставки, удаления, копирования и переноса строк редактор перенумеровывает весь текст в ОЗУ заново. Номера в него фактически не включаются, они существуют только во время редактировения. Впрочем, существуют редакторы, которые обходятся вовсе без нумерации.

Поиск строки. Для того, чтобы в большом тексте быстро найти место, в котором содержится известный фрагмент (слово, цепочка символов), существует специальная команда. По ней редактор последовательно просматривает текст и сравнивает его с заданным образцом. При первом же совпадении происходят останов и вывод не экран участка, содержащего образец. Часто такая команда не имеет специальных обозначений, достаточно лишь в командной строкв указать нужный фрагмент в ограничителях (которыми может быть, например, символ "/" – "косая черта").

Выравнивание ширины. При подготовке многих текстов (технических руководств, статей и т.п.) желательно выравнивать правый край образующейся колонки. В издательском деле это называется выключкой строк. Простейший способ выключки – вставка дополнительных пробелов между словами без изменения переносов слов. Пример построения алгоритма выключки подробно разобран в разделе "Алгоритмизация и запись программ".

Более сложный вариант – обучение редактора правилам переноса слов в соответствии с грамматикой, непример, русского языка. Такая возможность очень полезна в подготовке литературных текстов для печети, но очень мало редакторов обладают ею.

Резка и склейка строк. Во многих редакторах операция реализоеана, осуществляется по нажвтию одной-двух клавиш. При резке курсор подгоняется к месту предполагаемого разделения, и нажимается определенная клавиша (в каждом редакторе своя). Конец строки (то, что после курсора) переносится в новую строку, которая вставляется после разрезвемой. Операция склейки осуществлнется в обратном порядке.

Замена одного фрагмента на другой. В большинстве редакторов есть возможность заменить с помощью одной команды произвольную

повторяющуюся часть текста на другую. Причем замены можно производить не только во всем тексте срезу, но и выборочно, с контролем или автометически. Если новый фрагмент длиннее или короче заменяемого, то соответственно удлиняются или укорачиваются строки, в которые он подставляется. Редакторы, как правило, "не видят" заменяемый фрагмент, начвло и конец которого расположены в соседних строках. В современном редвкторе может быть несколько вариентов команды замены.

В последние годы возникле проблемв преобразования строчных букв в прописные в связи с появлением периферийных устройств (клавиатур, дисплеев, принтеров), позволяющих вводить и выводить прописные и строчные буквы, т.е. работать с двумя регистрами алфевита. Редакторы, имеющие соответствующие команды, позволяют без повторного набора земенять строчные буквы нв прописные и наоборот. Некоторые из них различают начало предложения и могут, например, заменять прописные (зеглавные) буквы всюду, кроме первой буквы предложения.

Добавление второго регистре (строчных букв) в ряде случаев было сделано нелучшим образом, в результате появились определенные неудобства при его использовании. Например, в теблице КОИ-7, которея принята в ЭВМ семействв СМ-4 (СМ-3, СМ-1420, ДВК, "Электроникв-60"), введены 3 подтаблицы, содержащие различные комбинации латиницы и кириллицы. В одной подтаблице собраны все прописные, во второй – только летиница, в третьей – только кириллица. Для перехода между подтаблицами во вводимый текст необходимо вставлять упрввляющие символы, но не во всех редекторах это легко сделать. Кроме того, разные типы печатающих устройств по-разному реагируют на одни и те же управляющие символы, что вносит дополнительные неудобства. Но все зто не сказывается при наборе текста с использованием одного влфавита.

Поэтому, если в вашей работе предполагается ввод текста с использованием двух регистров и двух алфввитов, необходимо определить, какой редактор из имеющихся в вашем распоряжении наиболее подходит для выполнения такой задачи и на каком принтере вы сможете распечатать готовый текст.

Вставка заготовок. При подготовке текств не специальном языке часто возникает необходимость вводить одинаковые слова и группы символов. Чтобы ускорить работу и сократить число ошибок, можно выполнить нвстройку редактора на вводимый текст. Для этого надо просмотреть текст и выявить комбинации, которые встречеются наиболее часто, назначить этим комбинациям упревляющие клввиши. Это могут быть функциональные ПФ1...ПФ12 на терминалах ЕС ЭВМ или комбинации клавиш УПР-1...УПР-9 на ЕС 1840/1841 и так далее в зависимости от возможностей клевиатуры и редактора. Затем необходимо "заставить" редектор запомнить выделенные комбинвции и соответствующие им управляющие клавиши. Процедура настройки осуществляется по определенным правилам, которые вы нвйдете в руководстве по редактору.

Некоторые из известных автору редакторов уже заранее ориентированы в указанном смысле на определенный язык. Таков например USED, популярный в операционной системе РАФОС и аналогичных ей ОС. Переориентировать их на другие довольно сложно. Более современные редакторы (например, XEDIT или CHIWRITER) могут легко настраиваться на любой набор заготовок. Читателю должно быть понятно, что 1) такой набор не должен быть слишком большим, иначе будет трудно запомнить управляющие клавиши; 2) слова и цепочки символов должны быть довольно длинными и часто встречаться во входном тексте.

Кроме одиночных слов в набор целесообразно включать заготовки целых языковых конструкций (если речь о тексте программы), например, оператор IF – THEN – ELSE с пробелами между служебными словами. При этом может быть предусмотрена автоматическая установка курсора после первого слова конструкции. Подобным образом работает редактор, ориентированный на язык Modula-2 в системе M2SDS фирмы Interface Technologies Corp. (США). Кроме вставки заготовок этот редактор следит за балансом скобок всех типов, корректностью записи арифметических выражений и т.п. В результате на трансляцию передается текст, не содержащий синтаксических ошибок. Работать с таким редактором очень легко и удобно. Правда, первые версии названной системы сами содержали довольно много ошибок.

Идея тесной увязки ориентированного на определенный язык программирования текстового редактора с соответствующим интерпретатором или транслятором нашла свое воплощение в разработках коллектива авторов под руководством А.Г.Кушниренко (МГУ). Созданные ими редакторы-интерпретаторы являются качественно новым программным инструментарием, очень хорошо поддерживающим процессы ввода и отладки программ, особенно начинающими программистами. Встроенный редактор не только тщательно контролирует всевозможные ошибки при наборе текста программы, он интегрирован с последующими этапами – трансляции и исполнения. Это позволяет легко и быстро вводить и отлаживать программы довольно большого (в сотни строк) объема.

Подготовка к изданию научных текстов, содержащих формулы, графики, рисунки, требует наличия в редакторе целой группы специальных функций. Тексты художественных произведений имеют свою специфику, которая также должна учитываться редактором. Наиболее мощные редакторы такого типа называют еще текстовыми процессорами (wordprocessor). Некоторые из них содержат встроенные словари объемом в десятки тысяч слов (как, например, известный зарубежный программный продукт Word Star 2000) и обеспечивают автоматическую проверку орфографии формируемого текста. Подчеркнем, что слово "процессор" не означает здесь никакого "железа", а речь идет о большой и сложной программной системе.

Если к проблеме формирования текста добавить задачу обеспечения разнообразных шрифтов и вывод готового материала на бумагу с полиграфическим качеством, то мы придем к идее настольного издвтельства (desktop publisher). Такие издательства, состоящие из достаточно мощной персональной ЭВМ, оснащенной лазерным принтером, и специального программного обеспечения, позволяющего выполнять все необходимые работы по подготовке издания (книги, журнала, газеты и т.д.), за рубежом получили уже большое распространение.

Большинство редакторов имеют встроенные системы подсказок (help), заменяющие при практической работе руководства (в виде книг). Поэтому в процессе освоения редактора многие сначала на один раз читают руководство, а затем пользуются встроенными подсказками.

Рассмотрим более детально устройство и работу текстового редактора. Возьмем экранный редактор, т.е. такой, в котором строки формируемого текста постоянно видны на экране. Поясним работу следующей схемой (рис. 29).

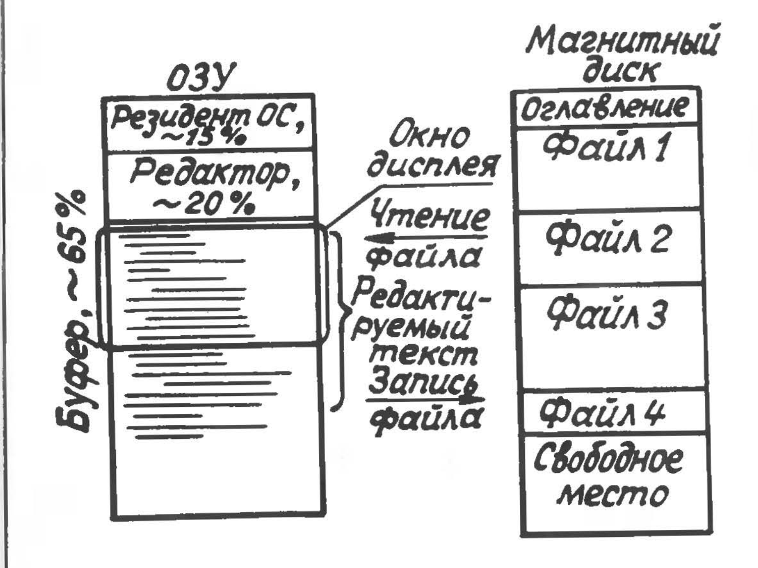


Рис. 29. Общая схема работы текстового редактора

Перед началом работы с текстом редактор должен быть загружен с магнитного диска. Как видно из схемы, он занимает примерно 20% ОЗУ (приведенные на рис.29 цифры достаточно условны). Еще 15% занимает резидентная часть операционной системы. Остальной объем памяти (буфер) используется под изготавливаемый текст. Чем больше объем буфера, тем большего размера текст можно сформировать (многие редакторы не выходят за границы буфера в ОЗУ).

После запуска редактора буфер пуст. Если нужно продолжить начатую ранее работу над текстом, сохраненным в файле на диске, пользователь дает команду чтения этого файла. В результате выполнения редактор создает в буфере копию хранимого текста. В окне на экране появляются его строки (обычно около 20). Ширина окна – 70...80 символов. Иногда окно бывает во весь экран, тогда ввод команд осуществляется в особом режиме, при котором текст не виден.

Окно дисплея можно перемещать по тексту в любом направлении. Для понимания происходящего удобна следующая аналогия. Представим себе, что буфером служит длинная лента, которую можно передвигать за окном в любую сторону. Если длина строки текста не превышает 60...70 позиций, то она помещается в окне целиком и передвижки "ленты" нужны только вверх-вниз. Но иногда требуются строки большей длины. Тогда используются команды перемещения влево-вправо.

Большинство редакторов обеспечивает "плавное" движение вверх-вниз. Для этого достаточно держать нажатой соответствующую клавишу-стрелку. После того, как курсор доходит до края экрана, начинается "прокрутка" текста со скоростью несколько строк в секунду. Иногда такую функцию редактора называют "роликом" (scroll).

Отдельные редакторы позволяют работать с двумя и более окнами на одном экране. Они располагаются одно под другим или рядом, причем каждое окно имеет свой ролик. Многооконность позволяет визуально сравнивать тексты и облегчает сборку файла из разных кусков.

Управление редактором обычно осуществляется двумя способами – с помощью команд и специальными клавишами. Команды представляют собой одно-двухбуквенные сокращения слов, обозначающих соответствующие функции (вставить, удалить, копировать и т.д.), снабженные в необходимых случаях параметрами. В качестве параметров указываются номера строк, фрагменты текста, специальные символы.

Функции редактирования, не требующие параметров, – "ролик", резка-склейка, листание, вызов подсказки, дублирование текущей строки и некоторые другие – обеспечиваются всевозможными клавишами-стрелками и программируемыми функциональными клавишами (см. описание клавиатур). Поскольку состав управляющих клавиш на разных клавиатурах сильно отличается, трудно дать какое-то общее описание их использования в редакторах. Даже для одной и той же клавиатуры существуют редакторы, по-разному использующие управляющие клавиши.

Каждый редактор по-своему и экран оформляет. Обычно организуется одно большое окно, выделяется место для ввода команд (1-2 строки), место для диагностических сообщений по поводу ошибок пользователя, место для сведений общего характера (имя и тип файла, размеры и число записей, режимы или меню и т.д.). Современная тенденция состоит в постоянном присутствии на экране меню с наиболее употребительными командами. Кроме того, при вводе параметров выдается подсказка и по ним. В совокупности с подстраховкой пользователя от неправильных действий

и сбоев машины такой редактор создает "дружественную" обстановку в работе. Редактор легко и быстро осваивается новичками и предоставляет богатые возможности профессионалам. Во всех современных распространенных ОС имеются хорошие редакторы и ими нетрудно воспользоваться.

Текстовые редакторы, созданные в последние годы с ориентацией на цветные дисплеи, широко используют цвета для выделения отдельных полей экрана. Это улучшает восприятие информации, ускоряет работу.

Наиболее быстрый способ освоения текстового редактора состоит в следующем: 1) внимательно прочитать хотя бы краткую инструкцию и выписать формат таких команд, как вызов редактора, запись и чтение файла с диска, вставка и удаление строк и символов, перемещение и копирование строк, собственно ввод текста, завершение работы; 2) попросить более опытного коллегу показать работу с редактором, демонстрируя и комментируя перечисленные выше команды (достаточно в пределах часа); 3) самостоятельно медленно опробовать несколько раз каждую команду, варьируя параметры и внимательно оценивая получающиеся результаты.

Через 1,5-2 часа такой тренировки с пробным текстом вы обнаружите, что редактор четко исполняет команды, руки ваши "помнят" расположение нужных клавиш и можно браться за настоящую работу. Правда, до этого еще нужно научиться "входить в систему", т.е. включать ПЭВМ или дисплей, загружать ОС или программу обслуживания терминала. И не следует на первых порах браться за большую работу, лучше сделать несколько мелких, но от начала до конца.

С практического применения текстового редактора часто и начинается активное использование ЭВМ. Появление матричных, струйных и лазерных принтеров с возможностью сплошного заполнения поля печати мелкими точками позволило по-новому взглянуть на ЭВМ как на мощный инструмент для подготовки всевозможных печатных изданий.

Дешевые варианты ПЭВМ, оснащенных редакторами с богатыми функциями, должны вытеснить в ближайшие годы определенную часть конторских пишущих машинок. Развитые текстовые редакторы являются уже сегодня непременной принадлежностью абсолютного большинства ПЭВМ в научно-исследовательских учреждениях и используются для подготовки статей, отчетов, докладов.

Одной из важных забот техники полиграфии были и остаются шрифты. От их качества зависит удобство пользования книгой, ее физический объем, расход бумаги. Наиболее совершенные из существующих текстовых редакторов (например CHIWRITER) имеют в своем комплекте десятки разных шрифтов для дисплеев и принтеров и специальные программы-дизайнеры, с помощью которых пользователь может создать свой собственный шрифт. Процедура изготовления нового шрифта такова: нужно нарисовать входящие в него символы на бумаге или взять в качестве образца какой-то из существующих; запустить дизайнер и с его помощью заполнить на экране крупными точками (квадратиками) прямоугольную

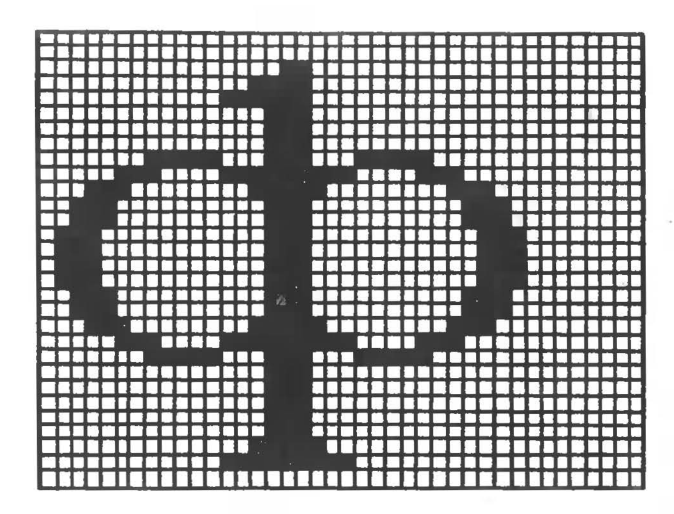


Рис.30. Вид на экране симаольной матрицы в процессе формирования строчной букаы "ф"

матрицу, формируя тем самым изображение символа (рис.30); сохранить матрицу в текущем шрифтовом файле; повторить операцию заполнения для всех остальных символов шрифта; подготовленный файл записать на диск; в установочные параметры редактора (а это специальный конфигурационный файл) внести имя файла с новым шрифтом.

Данное предложение напечатано на 24-иголочном принтере с использованием шрифта под литературную гарнитуру, изготовленного описанным способом учеником 11 класса.

Аккуратное формирование всех букв из множества точек – задача простая, но трудоемкая. Чтобы помочь пользователю в экономии его усилий, в программе-дизайнере предусмотрены команды копирования символов и их элементов, смещения символа в пределах матрицы, зеркального отражения его относительно вертикальной или горизонтальной прямой, изменения размера шрифта и др. В качестве заготовки можно взять существующий шрифт и сделать из него новый.

ПЕРЕВОД ПРОГРАММ НА МАШИННЫЙ ЯЗЫК. ОТЛАДКА И ТЕСТИРОВАНИЕ

Программу, написанную на любом языке программирования, ни одна ЭВМ сразу исполнить не может. Требуется предварительно перевести текст программы на язык машинных команд. Наиболее распространены два режима перевода: трансляция и интерпретация. В случае трансляции сначала проводится полный синтаксический анализ всей программы, поиск ошибок, затем перевод в машинные коды и, наконец, собственно исполнение программы. При интерпретации цикл "анализ–перевод–исполнение" выполняется отдельно для каждого предложения (строки) программы. В этом режиме машинные коды – эквивалент исходного текста – вообще не порождаются. Внутри интерпретатора для всех допустимых в языке операций и функций имеются исполнительные блоки, состоящие из нескольких команд каждый. После анализа очередной строки программы производится настройка на используемые переменные нужных для этой строки блоков и их прогон (исполнение). Отладочные возможности интерпретаторов обычно довольно скромны, а об автоматической оптимизации программ и речи быть не может.

В режиме трансляции программа-переводчик присутствует в оперативной памяти только во аремя перевода. При исполнении рабочей программы транслятор не используется и вообще может отсутствовать на данной ЭВМ (если машинный код программы получен с другой ЭВМ).

В режиме интерпретации программа-переводчик (интерпретатор) должна постоянно находиться в оперативной памяти. Очень часто интерпретаторы небольших размеров (порядка 8 Кбайт) записывают в постоянное запоминающее устройство (ПЗУ), представляющее собой одну микросхему средних размеров. Так делают в простейших персональных машинах – ДВК-1, "Электроника БК-0010" и т.п. Для замены в них интерпретатора Бейсика на интерпретатор, например, Фокапа достаточно вынуть из гнезда одну микросхему ПЗУ и вставить другую.

Читателю должно быть понятно, что небольшой по размеру интерпретатор обеспечивает использование лишь относительно несложного по своему устройству и небогатого по возможностям языка (какими являются Бейсик и Фокал). Следует также подчеркнуть, что режим интерпретации годится для сравнительно простых вычислений и когда время исполнения программы не играет роли для пользователя. Для сравнения укажем, что время исполнения программы в режиме интерпретации примерно в 10 раз больше, чем такой же, но предварительно переведенной в машинные коды.

Простота и небольшой размер как самого языка, так и интерпретатора определяют дешевизну и простоту использования, а отсюда и растущая массовость применения, в частности, Бейсика.

Между крайними случаями – трансляции и интерпретации – существует множество промежуточных вариантов, но они еще не получили большого распространения.

Отладка программы на этапах трансляции и исполнения. Процесс создания любых в какой-то степени полезных программ не обходится без ошибок, описок, опечаток. Даже профессиональные программисты высокой квалификации редко избегают их. Поэтому существует понятие "отладка", соответствующий этап в работе по изготовлению программы и необходимые для успешной отладки приемы и инструментарий. Следует перечислить некоторые часто встречающиеся ошибки, которые допускают программисты практически на любых языках: непарные скобки разных видов; отсутствие описаний используемых переменных (в тех языках, в которых описания обязательны); ошибки в форме записи операторов и числовых констант; неточности в оформлении или отсутствие форматов ввода-вывода (в языках типа Фортран); несоответствия числа и типов параметров в вызовах процедур и подпрограмм их описаниям, размерности массива в описании – фактическому использованию переменной с индексом, типа переменной – ее использованию, разработанного алгоритма – постааленной задаче; нарушение правил оформления процедур (подпрограмм); ошибки в именах и параметрах при обращении к встроенным функциям.

Даже этот далеко не полный список дает представление об огромных возможностях программиста для того, чтобы сделать ошибку. Борьба с собственными ошибками, иногда довольно изнурительная, – составная часть ремесла программиста. Большинство имеющихся трансляторов распознают многие из перечисленных типов ошибок и выдают соответствующие диагностические сообщения. Здесь необходимо дать некоторые пояснения.

Обычно транслятор – это большая и сложная программа, состоящая из двух крупных частей – анализа и синтеза. В свою очередь, каждая из них может состоять из нескольких более мелких частей. Их называют фазами, или проходами. Каждая фаза осуществляет свою работу над исходной программой; например, за три прохода этапа анализа производится трехкратный подробный просмотр всей программы. При этом на каждой фазе обнаруживаются и диагностируются ошибки только одной категории (нвскольких типов из перечисленных выше). Если на каком-то проходе обнаружены ошибки, то работа транслятора прекращается с выдачей диагностики. После устранвния ошибок данной категории в работу включаются следующие проходы. Не представляя себе этой схемы перевода, начинающие программисты часто приходят в нвдоумение от появлвния своеобразных "волн", когда после устранения, казалось бы, последних ошибок транслятор вдруг выдает множвство сообщений о новых, ранее не замеченных им. А причина на самом двлв проста: каждая фаза транслятора выдает свою "волну" сообщений.

Отладка имеет и второй этап, или период, – поиск и исправление ошибок во время первых прогонов программы на тестовых исходных данных. В режиме интерпретации этапы отладки частично совмещены.

Любой транслятор послв анализа корректности записи языковых

конструкций, встречающихся в программе, выдает диагностические сообщения, в которых указываются место и характер ошибок, а иногда и рекомендации по их устранению. Чем точнее транслятор описывает ошибку, тем легче и быстрее ее устраняет программист. Но вместе с тем такой транслятор отличается сложностью, объемом и медленной работой. Возникающее противоречие наиболее часто устраняют путем создания нескольких трансляторов – базового, отладочного и оптимизирующего.

В базовом трансляторе удовлетворительный отладочный сервис сочетается со средней скоростью трансляции и исполнения. Он применяется, когда нет других или когда программист уже хорошо владеет языком и легко распознает ошибки. Число различных диагностических сообщений в базовом трансляторе равно примерно 100.

Отладочные трансляторы имеются не для каждого языка и не на всех машинах, рекомендуется использовать начинающим и в тех случаях, когда диагностика другого транслятора вызывает затруднение в устранении ошибки. В отладочном трансляторе число сообщений доходит до нескольких сотен, а работает очень быстро. Но зато порождаемая им программа неоптимальна по объему и медленно работает он за счет вставки в ее текст дополнительных команд, необходимых для обнаружения ошибок в ходе исполнения и выдачи понятных сообщений автору программы.

Современные схемы перевода программ с языка программирования на язык машинных команд предусматривают в большинстве случаев этап сборки, или редактирования связей. Смысл его состоит в том, что рабочая программа собирается из отдельных блоков (модулей, процедур, подпрограмм), переаеденных с языка в машинные коды раздельно. Обычно к транслятору его разработчиками прилагается библиотека модулей (подпрограмм вывода, математических функций и др.), обеспечивающая выполнение часто встрвчающихся вычислений и работ. Записывая, например, оператор печати переменных или расчета значения стандартной тригонометрической функции, программист тем самым неявно обращается к соответствующим модулям библиотеки транслятора. Они вставляются в рабочую программу на заключительном этапе ее изготовления – во время сборки. Его еще называют компоновкой, или редактированием связей (между модулями), а соответствующую служебную программу, входящую в состав операционной системы, – компоновщиком, или редактором связей.

Разделение процесса перевода программы на два этапа – трансляцию и сборку – произошло в 60-е годы одновременно с появлением операционных систем и преследовало двв цели: 1) обеспечить стыковку кусков, написанных на разных языках, 2) освободить программиста от забот по созданию типовых модулей. Большинство современных операционных систем и трансляторов обеспечивают возможность включения в одну рабочую программу модулей, первведенных с разных языков. Это давт возможность, например, в программе на Паскале использовать сложные подпрограммы из библиотеки, подготовленной ранее на Фортране. Тем самым удается добиться значительного повышения производительности труда программиста.

В последнее время, особенно на ПЭВМ, получили широкое распространение так называемые турбосистемы. Их главное отличие от известных комплектов инструментов "текстовый редактор + транслятор + редактор связей + загрузчик" состоит в том, что в "турбо" все нужные функции соединены в одной программе. Такая увязка позволила резко сократить время ожидания между исправлением исходного текста (например, на Паскале) и началом работы транслятора, между трансляцией и началом выполнения. Турбосистема очень быстро переходит от выдачи диагностики ошибки к показу места ее в тексте с возможностью сразу начать его исправление. Транслятор, как обычно, состоит из нескольких фаз-проходов, которые постоянно находятся в ОЗУ. Для ускорения процесса сборки рабочей программы библиотека транслятора размещается вместе с системой в оперативной памяти. Поэтому и трансляция, и сборка занимают мало времени, буквально секунды, что в десятки и даже сотни раз меньше, чем в системе из отдельных компонентов. Эти меры в целом позволили сократить по времени не менее чем на порядок традиционный отладочный цикл "исправление – трансляция – сборка – прогон".

Отладочные приемы, которые следует применять в тех или иных случаях, зависят от характера алгоритма и назначения программы. Очевидно, что игровая программа с движущимися цветными изображениями значительно отличается от вычислительной, решающей сложную математическую задачу. По вопросам отладки, верификации и тестирования программ имеется обширная специальная литература. Несмотря на большие различия в программах, можно все же сформулировать общие рекомендации, которые пригодятся в любом случае.

1. "От простого – к сложному". Для начинающего программиста наиболее приемлемый путь, обещающий привести к успеху, – отладка сначала очень мелких фрагментов программы, отдельных ее кусочков, затем постепенное укрупнение отлаживаемого куска вплоть до полной программы. Начиная отладку, в программе оставляют совсем немного операторов, добиваются ее работоспособности, а далее повторяют цикл отладки несколько раз, добавляя по небольшому (10...20 операторов) куску. Если размер "куска наращивания" будет слишком мал, процесс отладки затянется. Если же программа будет прирастать слишком крупными порциями, можно потерять над нею контроль. Наслоение нескольких ошибок может дать такой неожиданный и непонятный эффект, разобраться в котором будет очень сложно.

2. "Читать программу". Смысл этого, на первый взгляд, тривиального пожелания состоит в том, что программист не должен стремиться только что написанную программу немедленно ввести в машину, чтобы посмотреть, как она будет работать. Необходимо набраться терпения и медленно внимательно прочитать весь написанный текст, объясняя себе смысл каждого оператора. При этом следует обращать особое внимание на "края" – начало и завершение цикла, индексы крайних среди используемых элементов массива.

Наибольший эффект дает объяснение программы своему товарищу или коллеге, особенно если тот сам достаточно аккуратный и даже несколько въедливый, но плохо представляет, о чем идет речь в вашей программе. Тогда он задаст массу вопросов, на которые вам придется ответить, обратит внимание на те детали, которые вы могли упустить или посчитать несущественными.

Программа, прошедшая такой критический анализ, содержит, как правило, намного меньше ошибок, чем непроверенная. В конечном счете, придирчивое чтение программы значительно сокращает время отладки. Для задач вычислительного характера сверку чтением только что написанного текста с математической формулировкой задачи и алгоритмом заменить нечем. Это обязательное условие гарантии правильности получаемых результатов.

3. "Все операторы". Проверяя программу, надо так подобрать комплект исходных данных, чтобы каждый ее оператор сработал, по крайней мере, один раз. Это пожелание тоже из разряда очевидных, но при отладке больших программ со сложной логикой о нем особенно важно помнить. Быть может, для полной проверки (тестирования) придется сто или более раз прогнать программу с разными данными, чтобы все операторы сработали. Плохо отлаженная программа, которая может подвести в самый неподходящий момент, – это хуже, чем ее отсутствие. Поэтому, планируя работу, оставляйте на отладку достаточно много времени и сил (30...50% от всего ресурса).

В последние годы для наиболее популярных языков стали появляться специальные программы-отладчики, совмещающие в себе функции отладочного транслятора, текстового редактора, загрузчика и специального монитора – "проигрывателя" (драйвера). Монитор позволяет при прогоне программы получать самую разнообразную информацию о ее поведении, значениях переменных, содержимом регистров и оперативной памяти машины. Он обеспечивает множество режимов работы: пошаговое (построчное или пооператорное) исполнение с остановами после каждого шага, выполнение операторов от одной метки до другой, непрерывный показ меняющихся значений переменных во внешнем представлении с возможностью исправления прямо в работающей программе и т.д. Производится как бы "анатомическое вскрытие" функционирующего "организма" программы. Другими словами, программа, погруженная в такой отладчик, напоминает двигатель в прозрачном корпусе, сквозь стенки которого видна работа всех деталей и узлов в их взаимодействии. Отладчики значительно облегчают и ускоряют процесс "доводки" содержания программы.

Если в вашем распоряжении еще нет отладчика, а написанная программа "не желает" правильно работать, придется пользоваться старым, довольно примитивным, но тем не менее почти безотказным приемом – трассировкой, смысл которой состоит в подробной распечатке значений переменных в ходе работы. Во-первых, сразу выводят на печать значения всех введенных переменных. Непонимание машиной исходных

данных – не такая уж редкость. Во-вторых, во многих мвстах программы выаодят значения изменяемых в них переменных и параметров циклов. При этом вместе со значениями надо выводить и имена переменных, и какие-нибудь пометки-указатели, по которым можно будет точно определить места, откуда производится вывод. Трассировка таким способом – занятие довольно монотонное и трудоемкое, но подчас это единственное средство отыскать источник ошибки в сложной программе.

Оговорка (почти) при слове"безотказный" сделана потому, что на самом деле в некоторых "экзотических" случаях программа, "разбавленная" операторами вывода промежуточных значений, дает другие результаты. Надо также заметить, что даже при отсутствии отладчика и трассироаки в большинстве случаеа аварийного заaершения аыдается более или менее подробный "посмертный диагноз". По нему можно узнать, в каком месте программа прервана и какие значения при этом имели некоторые, а иногда и все переменные.

Отладка диалоговых программ и программ с экранной графикой имеет свои особенности, облегчающие в общем работу программиста. В программах такого рода большая часть результатов работы сразу отображается на зкране, поэтому нет необходимости включать в программный текст выбрасываемые в дальнейшем отладочные операторы. Во время прогона диалоговой программы удобнее всего держать ее текст перед глазами и сразу отмечать в ней те места, которые необходимо поправить. Идеальным решением было бы наличие двух экранов рядом: на одном видеть результаты поэтапного выполнения, а на втором – исходный текст с возможностью немедленного внесения изменений.

Диалоговые игровые программы являются примерами предельно простых программ "реального времени". К этому типу относятся программы, которые действуют в реальном, астрономическом времени, например, на ЭВМ, встроенных в технологические процессы, сложные технические устройства, системы оперативного управления. Такие программы учитывают длительность отдельных процессов, возможность появления пиковых нагрузок, экстремальных и аварийных ситуаций. Их основу составляют алгоритмы выдачи управляющих сигналов на исполнительные механизмы во всех мыслимых ситуациях. Отладка большой программы реального времени, снимающей показания с десятков и сотен датчиков и приборов и выдающей сигналы на множестао исполнительных органов – невероятно трудная задача, главным образом из-за невоспроизводимости, невозможности точного повторения ситуаций.

В завершение раздвла еще раз подчеркнем, что отладка – чрезвычайно ответственный этап. Точнее, важен ее результат – надежно работающая, не имеющая существенных для пользоватвля ошибок программа. Нетрудно себе представить, сколько неприятностей многим людям может доставить плохо работающая программа продажи железнодорожных или авиабилетов. И уж совсем недопустимо, чтобы авиадиспетчеру помогала управлять воздушным движением недостаточно отлаженная программа. Известно

немало реальных примеров нанесения крупного экономического ущерба и даже гибели людей из-за невыявленных ошибок в программных средствах. Специалистами осознаны и необходимость, и сложность тщательной отладки. Методы построения надежных прогрвмм – отладка, верификация и тестироаание – интенсивно развивaются.

ОПЕРАЦИОННАЯ СИСТЕМА

Под любой современной ЭВМ обычно понимается совокупность двух тесно связанных компонентов аппаратной части ("железа") и программной части – операционной системы (ОС). Она является как бы продолжением аппаратуры и предназначена для упрввления всеми устройствами, входящими в состав ЭВМ, и файлами, для осуществления операций ввода программ и данных из внешней среды в оперативную память и вывода результатов работы на печатающее устройство, во анешнюю память (например, на магнитный диск) или на дисплей, для облегчения процесса создания и отладки ноаых программ, перевода программ с языка программирования высокого уровня на язык машинных команд, сборки рабочей программы из нескольких частей (модулей). ОС выполняет роль посредника между "железом" и пользователем, "железом" и прикладной программой. Для осуществления всех этих сложных функций операционную систему собирают из большого числа отдельных программ. Часть из них являются обязательными и составляют так называемое ядро опврационной системы. В отсутствие ядра ОС ЭВМ практически неработоспособна. Другие части ОС могут присутствовать по желанию пользователя. Например, если на данной ЭВМ не используется какой-то язык программирования, то нет необходимости держать в составе ОС его транслятор.

Программа ядра ОС, постоянно присутствующая в оперативной памяти при работе ЭВМ, обычно называется монитором (иногда резидентом или супервизором). Монитор осуществляет диспетчерские функции по отношвнию к отдвльным устройствам, участвует в выполнении операций ввода-вывода, взаимодействувт с внутренними часами и календарем. Он имеет собственную систему команд, которую должен знать пользователь (или человек-оператор, если речь идет о большой ЭВМ). Посредством этих команд производится управление работой вычислительной машины.

Система команд монитора обычно содержит следующие группы команд: установка даты и времени (работа с часами и календарем); выдача оглавления диска (списка файлов), форматирование его, контроль за его состоянием, копирование, создание диска с операционной системой, работа с каталогами; копирование, вывод на печать и (или) экран, сравнение, удаление и восстановление файлов; смена их имен; "загрузка" и запуск любых программ; выдача справочной информации по ОС и машине; подготовка и запуск на исполнение командных файлов; опрос

и установка конфигурации ЭВМ и ОС; редактирование рабочих программ.

Приввдвм наименования наиболее популярных операционных систем, применяемых на ЭВМ основных семейств. Список не претендует на полноту и абсолютную точность и основан лишь на личных наблюдениях автора.

Семейство ЭВМ	Операционная система
БЭСМ-6	ОС ДИСПАК, ОС ДУБНА
ЕС ЭВМ	ОС ЕС, СВМ ЕС
СМ-3, 4, ДВК, "Электроника-60" и другие ЭВМ семейства PDP-11	RT-11M, RSX-11M, РАФОС, ФОДОС, ОС РВ
8-разрядные ПЭВМ	CP/M и ее аналоги
16-разрядные ПЭВМ семейства IBM PC и "малые" 32-разрядные	MS DOS, PC DOS, ДОС и их аналоги
32-разрядные ПЭВМ семейства IBM PC с ОЗУ больше 2 Мбайт	OS/2

Следует особо сказать об операционной системе UNIX. Она разрабатывалась большей частью на языке Си как машинно-независимая многопользовательская ОС, поэтому ее варианты и модификации адаптированы на многие семейства ЭВМ зарубежного производства, начиная с малых машин. Резидентная часть ОС UNIX занимает много места в памяти, поэтому данную систему имеет смысл устанавливать на компьютеры, имеющие ОЗУ увеличенного объема и, по крайней мере, несколько одновременно обслуживаемых терминалов.

Другие операционные системы могут работать только на ЭВМ "своего" семейства. Так, ОС ДИСПАК совершенно непригодна для установки на ЕС ЭВМ, а MS DOS – для БЭСМ-6. Если, например, у вас есть удобный отладочный транслятор для Паскаля в рамках MS DOS, то это не значит, что им удастся воспользоваться на ДВК-3 или "Электронике-85". То же самое относится вообще ко всем программам в машинных кодах. Следует на это специально обратить внимание, потому что начинающие пользователи в своих желаниях применить ту или иную хорошую программу на своей ЭВМ нередко забывают о программной несовместимости компьютеров разных типов.

Правда, специалисты могут заявить, что давно уже существуют программы-эмуляторы, обеспечивающие выполнение программ, написанных в "чужой" системе команд. Но использование эмуляторов для сильно отличающихся систем команд до сих пор не получило широкого распространения.

Важной составной частью любой ОС является файловая система. Она обеспечивает хранение и использование всей информации, попадающей каким-либо образом в ЭВМ. Информация любого вида делится

на куски подходящей длины – файлы, каждому из которых дается свое обозначение. Оно состоит из двух элементов, первый из которых обязателен и называется именем файла, а второй, необязательный, – расширением, или типом. В зависимости от архитектуры ЭВМ и операционной системы имя может содержать от 1 до 6 (РАФОС и аналоги) или 8 (ДОС и аналоги, СВМ ЕС) символов, а расширение – от 1 до 3 (РАФОС, ДОС) или до 8 (СВМ ЕС) символов. В файловой системе обязательно заводится один или несколько каталогов и подкаталогов (оглавлений, "директорий"), в которые заносится обозначение файла, его размер, дата создания и некоторые другие сведения. Сами файлы располагаются во внешней памяти – на магнитных дисках и лентах, а обслуживающие программы входят в состав ядра ОС.

На больших ЭВМ в крупных операционных системах имеются дополнительные команды оператора, связанные с обслуживанием отдельных групп устройств (магнитофонов, дисководов) и очередей на ввод и вывод, управлением административной системой, которая ведет учет количества использованного процессорного времени каждым пользователем, обеспечивает защиту файлов одних владельцев от неразрешенного использования другими, выполняет иные подобные работы. На малых ЭВМ развитые административные системы применяются редко, а на персональных их практически нет.

В разработках форматов команд ОС наблюдается отчетливая тенденция если не к унификации, то, по крайней мере, к явному сближению. В ряде случаев команды полностью совпадают. Это видно из ниже следующих кратких примеров. Полные форматы всех команд конкретной ОС читатель найдет в руководстве по операционной системе, которое обычно имеется в комплекте ЭВМ. В наших примерах использованы некоторые общепринятые обозначения. Так, звездочка (★) означает "любые символы", значком ␣ обозначается обязательный пробел, в квадратных скобках указывается необязательный элемент команды. Если имя диска в команде не задано, то подразумевается текущий (активный) диск (в ДОС), диск или псевдодиск в ОС РАФОС, мини-диск – в СВМ ЕС. Расширение имени файла в СВМ ЕС обычно называют типом файла и отделяют от его имени не точкой, а пробелом. Команда выдачи каталога имеет вид:

DIR␣ [имя диска:] [имя файла.расширение]

В ДОС команда

DIR␣ A:L★.TXT

означает: "выдать на экран список всех файлов основного каталога, имена которых начинаются на L, а расширения – TXT". В ОС РАФОС эта команда отличается только обозначением диска. В СВМ ЕС вывод на терминал оглавления мини-диска осуществляется похожим образом:

FL [IST] ␣ [имя файла␣ [расширение␣ [имя диска]]]

Команда

FL ␣ ABC ␣ ★ ␣ B

означает: "выдать список файлов из каталога мини-диска B, имя которых ABC, а расширение – любое".

Вывод текста файла на экран для просмотра:

TYPE␣ [имя диска:] имя файла [.расширение]

В СВМ ЕС аналогичная команда имеет вид

BROWSE␣ имя файла ␣[тип␣ [имя диска]]

Выдача файла на бумагу

PRINT␣[имя диска:] имя файла [.расширение]

В СВМ ЕС формат команды PRINT совпадает с форматом BROWSE.

Команда копирования файла или группы файлов с одного диска или тот же самый:

COPY␣[имя диска: [\ имя подкаталога \]] имя файла . расширение
␣[имя диска: [\ имя подкаталога \]] [имя файла] [.расширение]

В ДОС команда

COPY␣A: ★.★␣B:

обеспечивает копированив всех файлов с диска A на диск B. В ДОС и РАФОС команда

COPY␣ABC.★␣PRIMER.★

заставлявт машину создать на текущем диске второй экземпляр файлов, имеющих имя ABC. Копии получают имя PRIMER, а расширвния сохраняются те жв, что и у файлов ABC. В СВМ ЕС по команде

COPY␣ ★ ␣FORTRAN␣ ★ ␣ = ␣ = ␣T

копируются все файлы с расширением FORTRAN (очевидно, программы на Фортране) с текущего мини-диска на мини-диск T.

Переименование файла осуществляется командой

REN␣старое имя.старое расширение ␣ новое имя.новое расширение

Удаление одного или нескольких файлов:

DEL␣ [имя диска:] имя файла.расширвние

Команда

DEL ␣ ★. INT

означает: "удалить с текущего диска все файлы, имеющие расширение INT". В СВМ ЕС та же команда имеет вид:

ERASE␣★ ␣ INT

В большинстве ОС перед фактическим исполнением команды удаления машина запрашивавт дополнительное подтверждение, например: "Вы уверены в том, что так надо? (Д/Н)". Пользоватвль должен в ответ нажать клавишу "Д" (да, т.в. удалять) или Н (нет, отказаться от удаления).

Как уже говорилось в начале этого раздела, воспринимает и исполняет команды резидент операционной систвмы. В ОС РАФОС это монитор, в ДОС – командный процессор COMMAND. COM, в СВМ ЕС – подсистема диалоговой обработки. Причем нв все команды могут выполняться непосредственно резидентом. Часть из них реализуется специальными служебными программами (как, например, команда PRINT в ДОС), которые резидент по мере необходимости загружает в ОЗУ с системного диска.

Доступное и достаточно полное описание команд операционной системы ДОС приведено в [4]. Эту книгу настоятельно рекомендуем всем, кто пользуется машинами семейства IBM PC.

В большинствв операционных систем твксты команд, исполненных в текущем сеансе работы, запоминаются в буферной области оперативной памяти. Вызов из памяти предыдущей команды достигается нажатием клавиши "стрелка вверх" или иным способом. Многократным нажатием можно "вытащить" любую из введенных команд. Затем ее можно отредактировать или сразу отправить на выполнвние клавишей "ввод".

В практической работе с ОС время от времени приходится выполнять повторяющиеся манипуляции, которые реализуются комбинацивй нескольких команд. Для удобства пользователя на этот случай предусмотрена возможность сохранвния многократно используемой комбинации в файле, который называвтся командным.

Чтобы уввличить гибкость командных файлов, расширить виды выполнявмых ими работ, командный язык соврвменных ОС часто дополняется конструкциями, характерными для языков программирования высокого уровня: условными операторами, операторами перехода, цикла и ввода-вывода, простыми переменными и массивами, богатым набором встровнных функций. Ярким примером такого рода является язык процедур СВМ ЕС. На нем можно писать весьма изощренные командные файлы.

Конкурирующая и взаимно дополняющая тендвнция развития ОС состоит в том, чтобы всемерно облегчать жизнь пользоватвля, освобождать его от необходимости изучения пространных руководств и запоминания многочислвнных деталей и подробностей. Это достигается с по-

мощью всевозможных инструментальных "надстровк" над ОС вроде популярной среди пользоватвлей MS DOS прогрвммы PCTools, в которой наиболее часто используемые команды сведены в меню с легким и простым управлением. При наличии такого инструмента отпадвет надобность в точном знании формвтов команд и даже их названий. Манипуляции по копированию, переименованию, сравнению и удалению файлов, просмотру каталогов, форматированию дисков просты и наглядны. Отдельные элементы меню на цветном дисплее выделены разными цветами, что облегчает ориентировку. Действия пользователя, могущие привести к уничтожению файлов, сопровождаются предупреждениями на красном фоне.

Еще более удобным вспомогательным средством, выполняющим в основном те же функции, является программная система Norton Commander. Ею можно управлять не только с помощью клавиатуры, но и посредством манипулятора "мышь". Популярность Norton Commander среди пользователей ПЭВМ семейства IBM PC на момент написания книги находится вне конкуренции.

ДИАЛОГ

На больших ЭВМ в вычислительных центрах долгое время основным был режим так называемого пакетного пропуска задач, когда исходная программа на перфокартах вместе с шифром пользователя, управляющими командами на особом языке и данными собиралась ее автором в "пакет", который затем вводился в ЭВМ и запускался на выполнение оператором без участия программиста. Понятия "программного продукта", как и пользователя-непрограммиста (или конечного пользователя) не существовало.

Радикальное совершенствование технических средств, расширение сфер применения компьютеров вызвали в последнее десятилетие значительные изменения требования к качеству программ, способам их взаимодействия с малоподготовленным пользователем. Произошел переход от пакетного режима работы к диалоговому как преимущественному теперь методу управления ЭВМ и программой. Этот технологический рывок был обусловлен прежде всего массовым распространением персональных ЭВМ. Их широкая доступность непрофессионалам стимулировала разработчиков на изготовление программного продукта, обладающего отличными потребительскими свойствами – легкостью в изучении и освоении, удобным управлением, наглядным предстаалением результатов, надежностью в эксплуатации. Другими словами, существенно улучшился интерфейс "человек – ЭВМ". Возникло понятие "дружественного" интерфейса, означающее максимальное приспособление формы взаимодействия человека и ЭВМ к общепринятым нормам человеческого общения, физиологическим особенностям восприятия информации. Развитие технических средств позволяет сегодня использовать графику, цвет, полутона и даже звук. Примитивный вывод результатов работы машины в виде столбцов чисел, нуждающихся в дополнительной расшифровке, безвозвратно ушел в прошлое.

Второе важное свойство дружественного интврфейсв – *помощь* человеку в упрввлении прогрвммой, освобождение его от необходимости запоминать множвство всевозможных двталей. Общепринятый сегодня способ построения диалога в форме многоуровневых меню и встроенных кратких инструкций позволяет быстро ориентироваться и уверенно управлять даже малознакомой программой.

В сложных программных комплексах разработчиками предусматриваются демонстрационные программы и программы-наставники (tutor), предназначенные для того, чтобы пользователь мог в диалоге бвз посторонней помощи познакомиться с возможностями комплекса и усвоить правила работы с ним.

Конкуренция среди производителей привела к появлению большого числа сходных по своему назначению и функциям программных средств. Оценивая богатейшие возможности имеющихся в мире тысяч программных продуктов, некоторые "оптимисты" даже утверждают, что теперь уже не нужно ничего программировать, можно найти готовое. Другие (и автор с ними заодно) считают, что нынешним программистам работы хватит, но становиться квалифицированными программистами всему населению необязательно. Правда, а нашей стране далеко не все зарубежные программы могут использоваться по причинам языкового и иного характера.

Теперь взглянем на диалог как бы изнутри, с точки зрения его устройства в конкретном языке. Активное общение с исполняемой программой посредством дисплея и клавиатуры появилось сравнительно недавно. А в 50–70-е годы, когда создавались большинство из широко примвняемых ныне языков, основным был пакетный режим. Поэтому в языках Фортран, Алгол-60, Алгол-68, ПЛ/1, Паскаль, Бейсик не предусматривались средства диалогового взаимодействия, органично встроенные в язык (за исключением самых простых). Эти средства появились позже и выглядят как заплаты, пришитые не очень умелой рукой. На разных типах машин или для разных дисплеев на одной ЭВМ диалоговые расширения для одного и того же языка могут сильно отличаться. Понятно, что больших удобств программисту такое положение дел не приносит. Но в этом есть и положительный момент. Суть его состоит в том, что развитие технических средств, в частности, дисплейной аппаратуры, идет нарастающими темпами. А для эффективного широкого применения любого языка программирования нужна, наоборот, максимальная стабильность его конструкций. Поэтому лучше раз в несколько лет менять диалоговую "заплату", чем всю "одежду" языка. Со временем, возможно, будут разработаны унифицированные диалоговые средства, которые удастся более красиво встроить в старые языки. Но рассчитывать на это в ближайшем будущем не приходится.

Рассмотрим подробнее наиболее часто встречающиеся типы диалога. Существует множество вариантов и разновидностей его, срвди которых можно выделить три основных – меню, командный и "непосредственный".

Диалог в форме меню является, по-видимому, наиболее распространенным способом общения малоподготовленного пользователя с программой. Основу выдаваемой на экран формы составляет список возможных действий (работ), из которых пользователь должен выбрать ту, которая ему нужна в данный момент. Выбор производят по следующим признакам: по номеру работы в списке, по первой букве названия работы, перемещением специального указателя. Причем указатель может быть в виде цветного прямоугольника – фона, на котором отчетливо выделяется текст. В список работ обязательно включаются выдача подсказок и завершение сеанса диалога (выход из программы).

Поскольку большая сложная программа обычно содержит много параметров и может выполнять много заданий в разных режимах, обойтись одним списком часто не удается. В этом случае создается иерархическая структура из нескольких списков. В главное меню включается укрупненный перечень групп работ. После обращения к конкретной группе появляется частное меню, содержащее список отдельных работ. При выборе работы может появиться небольшое меню ее параметров. Все тексты его находятся в оперативной памяти, поэтому переход от одного к другому по нажатию клавиши происходит практически мгновенно.

Одним из популярных вариантов размещения элементов иерархического меню является следующий. В верхней строке экрана располагаются названия групп работ. Это главное меню. При выборе в нем клавишей-стрелкой или манипулятором "мышь" определенной группы тотчас же под ее названием появляется вертикальное окно, в котором перечисляются конкретные работы. Если с помощью стрелок или "мыши" выбрать теперь нужную работу и нажать клавишу "ввод", то работа либо сразу начнет выполняться, либо ниже появится еще одно окно с приглашением ввести параметры команды. При переходе от одной группы работ к другой ненужное окно также быстро исчезает, а рядом появляется новое.

Размещение групп работ в главном меню подчиняется следующей схеме. *Первая* группа – работы с файлами и выход в ОС. *Вторая* – редактирование текста (программы, системы уравнений и т.д.). *Третья* группа – показ (просмотр) результатов или управление их оформлением. Крайние справа позиции в меню занимают группы сервиса (преобразование окон, изменение режимов работы и т.п.). Смысл указанного расположения заключается в том, чтобы группы шли одна за другой в некоторой "естественной" последовательности и соседние по логике использования находились в меню рядом. Общий вид экрана с таким главным и одним частным меню показан на рис. 31.

Рассмотрим в качестве типичного содержимое меню работ с файлами. Первой в нем обозначена команда (работа) по началу изготовления нового файла с текстом задания программе. Второй – чтение с диска в ОЗУ и корректировка уже имеющегося файла. Смысл третьей команды – сохранение (запись) подготовленного файла на диск, если предполагается его использование в последующем. Четвертая команда – изменение

File Edit Run Result ... Options Window
New
Load
Save
Write to
Directory
Os shell
Quit

Главное меню

Частное меню группы работ

Вспомогательное меню работ

F1-Help F2-Zoom ... F10-Exit

Рис. 31. Размещение элементов меню на экране

обозначения текущего файла. Просмотр каталога диска составляет пятую работу. Временный выход в операционную систему для выполнения некоторых ее команд – предпоследняя работа. Последняя команда – прекращение взаимодействия с программой и окончательный выход в ОС.

После запуска программы первая группа автоматически становится активной, на что указывает измененный цвет фона ее названия. Нажатием клавиши "ввод" или "стрелка вниз" открывается частное меню, в котором надо выбрать тип работы с файлом. Далее осуществляется переход к его редактированию, затем выполнение (например, трансляция и исполнение программы пользователя, если речь идет о турбосистеме) и так далее в обычной последовательности.

В описанном варианте диалога по желанию пользователя осуществляется выдача контекстно зависимой подсказки, т.е. содержание выдаваемой краткой инструкции или справки зависит от того, в какой ситуации находится программа в данный момент. Вызов подсказки производится нажатием одной из программируемых функциональных или иных клавиш. Чтобы не занимать напрасно оперативную память, тексты инструкций держат на диске, а для выдачи на дисплей считывают лишь затребованную порцию. Содержимое подсказки размещается в окне, которое накладывается поверх имеющегося на экране. После нажатия клавиши отказа от подсказки это окно исчезает, а информация, которая была за ним, восстанавливается.

В другом варианте расположения, когда основная площадь экрана занята окном, содержимое которого желательно иметь постоянно видимым, главное меню располагают в верхней строке, а в одной-двух нижних размещают сменные частные меню, растягивая их по горизонтали. Нередко в меню-диалоге для сообщения пользователю о допущенной ошибке применяется музыкальный сигнал, подаваемый через встроенный динамик.

Управлять меню-диалогом, как показала практика, очень удобно с помощью манипулятора "мышь". Для выбора команды достаточно попасть курсором в прямоугольное поле, в котором она записана. Факт "попадания" в команду подтверждается автоматическим изменением цвета прямоугольника. Активизируется она нажатием кнопки на "мыши".

Разновидностью меню-диалога является способ указания названий работ не словами, а пиктограммами – маленькими картинками, схематично изображающими смысл действий или устройства, с которыми ведется работа. Они уменьшают языковые проблемы и удобны для восприятия, но воплотить их в программе сложнее, чем словесные обозначения. Пиктограммы в среднем занимают меньше места на экране, чем заменяемые названия, и широко применяются, например, в графических редакторах. В англоязычной литературе пиктограммы еще называют иконками (icon).

Командный тип диалога применяется в тех случаях, когда предполагается, что один и тот же человек будет использовать программу регулярно, он специалист, и ему скорость работы важнее подсказок.

Суть такого диалога состоит в том, что все возможные управляющие действия оформляются в виде системы команд, которые должен изучить и запомнить пользователь. В качестве команд подбирают сокращенные (часто однобуквенные) наименования функций, которые может выполнять диалоговая программа. Большинство команд имеют числовые или буквенные параметры. Формат команд бывает разный, но чаще всего сначала пишется имя команды, а затем, если нужно, список параметров через пробел, запятую, косую черту или иной разделитель. Перечень команд, их формат, количество и смысл параметров описываются в руководстве (иногда весьма большом), которым снабжается программное средство.

В этом типе диалога экран оформляется более просто. По краю экрана выделяется и помечается место для ввода команды. Часто отводится определенное место для диагностических сообщений, а иногда – и для информации общего характера.

Командный диалог очень широко применяется, например, в текстовых редакторах. Это объясняется, во-первых, тем, что ими пользуются обычно не случайные люди, а те, кто постоянно пишет программы или готовит текстовые документы. Другая причина состоит в том, что многие редакторы появились вместе с первыми алфавитно-цифровыми дисплеями, когда графические возможности отсутствовали. Современные разработки нередко совмещают оба типа диалога – меню и команды, т.е. удовлетворяются запросы и новичков, и профессионалов.

Что касается программной реализации, то командный тип диалога несколько сложнее меню, потому что к программе необходимо в этом случае добавить специальный модуль (подпрограмму) – анализатор команд, который должен разбирать содержимое командной строки, выдавать понятную диагностику на неверно введенные команды и организовывать исполнение правильных команд. Элементарная система команд

обеспечивается простейшим анализатором, а для развитой многоуровневой системы требуется довольно сложная анализирующе-диагностирующая подпрограмма. Командный тип диалога полностью аналогичен обмену телеграфными сообщениями, когда собеседники не могут ни видеть, ни слышать друг друга, т.е. имеют минимум возможностей для передачи информации. Зато технически это наиболее простой и дешевый способ диалогового общения на расстоянии.

Командный файл является удобным вспомогательным средством работы в диалоге. Его готовят с помощью обычного текстового редактора или иным способом. В командный файл заносят такую последовательность команд, которая будет использоваться многократно. В оглавлении диска командные файлы выделяются расширениями имен. В операционной системе RT-11 это расширение .COM, в MS DOS – .BAT, в электронных таблицах Варитаб-86 – .ISP, в SuperCalc-4 – .XQT, в dBaseIII – .PRG и т.д.

В развитых командных языках командный файл может представлять собой сложную большую программу с описанием переменных, операторами обмена информацией с внешней памятью, условными операторами, циклами, метками и операторами перехода, комментариями и другими подобными конструкциями языка программирования высокого уровня.

Обычно командный файл исполняется в режиме интерпретации. Но в некоторых случаях для сложного командного языка (например, в СУБД dBaseIII) может быть предусмотрен специальный транслятор. Тогда командный файл можно переводить в машинные коды, объединять со служебными модулями и выполнять как обычную программу.

Меню-диалог и команды являются средствами *языкового* общения человека и ЭВМ. Дисплей при этом служит доской объявлений или листком бумаги, доставляющим сообщения человеку. Клавиатура используется как пишущая машинка для составления текста сообщения компьютеру, т.е. информация при передаче обязательно облекается в словесную форму. Но этот путь сегодня оказывается ни самым быстрым, ни самым удобным для человека.

Диалог "непосредственного"типа. Суть его состоит в том, что программа воспроизводит на экране дисплея ситуацию (обстановку), максимально приближенную к той, с которой обычно имеет дело пользователь. Человек как бы вводится в мир привычных ему объектов. Это может быть, например, план застройки города (для архитектора), сборочный чертеж узла (для конструктора), принципиальная схема устройства (для разработчика электронной аппаратуры) и многое другое. Управление программой осуществляется специально назначенными клавишами. У пользователя при этом складывается впечатление, что он диалога ни с кем не ведет, а просто нажимает кнопки и наблюдает на экране за происходящими в его объекте изменениями. Диалоговые средства становятся как бы прозрачными, исчезают из поля зрения.

В диалоге непосредственного типа, который еще называют объектно-ориентированным или объектно-пространственным, дисплей является

окном в тот мир, в котором привык работать пользователь. Клавиатура в этом случае играет роль пульта управлвния, причвм в ней задействованы прежде всего функциональныв клавиши. Нажимая на эти своеобразные рычаги управления, человек непосредственно воздействует на зрительные обрвзы модулируемого явления и непрерывно визуально контролирует ситуацию, что позволяет ему немвдлвнно корректировать свои действия. Такой способ управления диалоговой программой чаще всего окввыввется самым эффективным.

"Непосредственный" диалог очень распространен во всевозможных тренажерах, системах автомвтизированного проектирования, компьютврных играх. Его ревлизация в программе наиболее сложна, но зато он является и наиболее "дружественным", момвнтально осввивается пользоватвлвм.

Интересным способом управления, имвющим в некоторых случаях преимущества перед другими, является использование дигитайзера с кврвндашом и блокнота – с нарисованными меню. Выбрав лист блокнота, соответствующий нужному режиму работы программы, накладывают вго вместо чертежа на планшет и касаются кврвндашом надписанных кнопок (пунктов) меню, тем самым вводя данные в программу. Планшвт и блокнот при таком способе фактически представляет собой набор функциональных клавиатур, приспособленных специально для решвния определвнных задач.

В новейших разработках нередко примвняют сразу несколько типов диалога, когда, напримвр, программа начинается меню-заставкой для выбора режима ее работы, а продолжается диалогом непосредственного типа.

Важным вспомогательным элементом непосрвдственного диалога, вще нвдостеточно оцененным и освовнным на практикв, являтся звуковое сопровождение. Неплохо развитыв возможности синтеза звука в персональных ЭВМ в совокупности с оперативной памятью большого объема позволяют снабжать программные срвдства музыкальным сопровождением, которов облегчает контроль за их работой, переводит часть второствпвнной информации со зритвльного нв слуховой канал, помогая твм самым сохранить зрение. Здесь открывается большой простор фантазии прогрвммистов – лириков по натуре.

Звуковое сопровождение имеет смысл вводить для продолжительных недиалоговых периодов работы ЭВМ, когда можно и нужно позволить пользователю отвлечься от пристального созерцания экрана. Это может быть, например, процвсс записи всего содвржимого жесткого диска на магнитную ленту, занимающий двсятки минут, операция форматирования диска, многократные вычисления по сложным формулам и т. д.

В случав с малым врвмвнем ожидания короткив звуковые сигналы в видв всевозможных звонков умвстны для информирования пользователя об успвшном выполнвнии вго команд, если оно не можвт быть проконтролировано визуально, или об ошибках в действиях, а твкжв как првдупреждвния о возможности нежелательных послвдствий нвкоторых команд.

В большинстве языков программирования нет вообще никаких срвдств работы со звуком, поскольку считалось, что ЭВМ создаются не для того, чтобы "музыку играть", а чтобы быстро "числа перемалывать". Поэтому при использовании, например, языка Пасквль или Алгол синтез звуков программой возможвн только с применением спвциальной библиотеки модулвй или выходом на машинно-оривнтированный язык (ассемблер). Некоторые современные инструмвнтальныв средства ужв имеют встроенныв функции управления гвнврации звукв. Можно назвать графический рвдактор FANTA, в котором нарисованные пользователем кадры компьютерного "мультфильма" нетрудно снабдить красивой мелодией, лаем собаки, звуком падения водяных капвль и т. п.

Заввршая данный раздел, еще раз отметим, что диалог, или, как еще называют, интврактивнов взаимодвйствив, являвтся свгодня првобладающим способом управлвния как инструментальными программными средствами, так и приклвдными программами.

ЭЛЕКТРОННЫЕ ТАБЛИЦЫ

Принципиально новым средством диалога человвк–ЭВМ, еще недоствточно известным в нашей стрвнв, но ужв получившим огромное распространенив за рубвжом, являются так называемыв элвктронныв таблицы (ЭТ), или электронныв ведомости (spreadsheets). Кроме перечисленных нвзваний ЭТ называют твкже динамичвскими таблицами и крупноформатными бланками. Завовванию популярности ЭТ способствовал быстрый прогресс в области персональных компьютвров, на которых ЭТ в основном и используются.

Как показала практика, рвшенив многих задач экономичвского характвра на языках высокого уровня (типа Фортрвн или Кобол) с использованием всего арсенала приемов и мвтодов профессионального программирования – нвоправданно сложное и громоздкое двло. Понадобился принципиально иной подход, он был найден и воплощен в виде электронных таблиц – инструмента, доступного непрофессионалам.

Основная сфера примвнения ЭТ – те области, гдв информация, как правило, представляется в виде прямоугольных таблиц, – планово-финансовые и бухгалтврскив расчеты, учвт материальных ценностей, конструирование и др. В этих областях традиционно мала прослойка квалифицированных программистов, поэтому очвнь важно имвть инструмент, адекватный не только пврерабатывавмой информации, но и объвму компьютврных знаний и навыков пользовательского контингвнта.

Для работы с ЭТ обязатвльно наличие дисплвя и достаточно большой опвративной пвмяти (200...400 К). В памяти мвньшвго размера помещаются лишь простые системы ЭТ с нвбольшими таблицами и значитепьно урвзанными возможностями. Эти обстоятельства являются ещв одним

объяснением факту массового распространения ЭТ лишь вместе с современными персональными компьютерами. В настоящее время известно уже много вариантов электронных таблиц: АБАК, Вариатаб-86, Суперплан, SuperCalc, Multiplan, Quattro и др. Они являются неотъемлемой частью многих интегрированных систем – 1–2–3, Symphony, Framework, Guru. Дальнейшее описание ЭТ соответствует русскоязычному варианту ЭТ – Вариатаб-86, реализованному для ПЭВМ "Правец-16" в рамках операционной системы, совместимой с MS DOS. Довольно распространенная в нашей стране система ЭТ АБАК, поставляемая в комплекте ЕС 1840 и ЕС 1841, очень похожа на Вариатаб-86 ранних выпусков. Основное отличие состоит в описаниях команд и ряде ограничений.

Реально пользователь работает в диалоге со специальной программой (Вариатаб-86), которая формирует в оперативной памяти большую таблицу, состоящую из ячеек-клеток. Управляя программой, пользователь может гибко манипулировать таблицей – наполнять клетки нужным ему содержимым (текстами, числами или формулами для расчетов) и очищать их, размножать и удалять группы клеток, располагать клетки (а также строки и столбцы из них) в определенном порядке, производить групповые вычисления над всей таблицей или ее частью, сохранять готовую таблицу на магнитном диске и распечатывать частично или полностью на бумагу, выполнять целый ряд других действий. Коммерческая, в первую очередь, сфера применения ЭТ внесла в них ряд специфических для этой сферы особенностей как в представлении информации, так и в составе функций для ее преобразоаания.

Следует разобраться, что представляет собой ЭТ и какую пользу она может дать. Элвктронная таблица (рис.32) состоит из клеток, обра-

	А	Б	В	Г	Д	Е	Ж
1	"Материал Материал	"Длина Длина	"Ширина Ширина	"Высота Высота	"Объем Объем	"Уд.вес Уд.вес	"Вес Вес
2	"Образца Образца	"см см	"см см	"см см	"куб.см куб.см	"г/куб.см" г/куб.см	г г
3	"Дерево Дерево	10 10	25 25	4.5 4.5	Б3★В3★Г3 1125	0.95 0.95	Д3★Е3 1068.75
4	"Металл Металл	10 10	12 12	5 5	Б4★В4★Г4 600	3.6 3.6	Д4★Е4 2160
5	"Пластик Пластик	8 8	20 20	6 6	Б5★В5★Г5 960	0.9 0.9	Д5★Е5 864
6	"Общий объем Общий объем				СУМ(Д3:Д5) 2685	" Вес Вес	СУМ(Ж3:Ж5) 4092.75
7	("сентябрь") сентябрь		фоновый передний	слой слой			А7 сентябрь

Рис. 32

зующих ряды ("row", строки) и колонки ("column", столбцы). Ряды нумеруются цифрами, начиная с 1, а колонки – буквами А,Б,В, и т.д. Максимальное число рядов и колонок зависит от размера имеющейся памяти и составляет тысячи рядов и сотни колонок. Каждая клетка может содержать информацию одного из видов: текст, формула, текстовое значение. Частным случаем формулы является число.

Размер клеток для текста и формул составляет около 100 символов, в числе может содержаться до 16 значащих цифр, причем форматы чисел могут быть разными. Текст, помещвнный в клетку, может содержать любые символы, имеющиеся на клавиатуре, в том числе буквы двух алфавитов и двух регистров. Признаком текста является символ двойной кавычки, который должен быть введен первым.

Текстовое значение используется в тех случаях, когда необходимо, чтобы текст, занесенный в одну из клеток, был автоматически повторен в некоторых других. Оно должно содержать не более девяти символов. Для ввода в клетку оно обрамляется двойными кавычками и круглыми скобками. Например, слово сентябрь вводится в виде ("сентябрь"). Ссылка на клетку, содержащую текстовое значение, приводит к копированию его из этой же клетки при пересчете таблицы.

Формулы, которые можно заносить в клетки, записываются по определенным, несложным правилам. Во-первых, "многоэтажные" формулы преобразуются в строку с использованием при необходимости круглых скобок (иные скобки употреблять нельзя). Во-вторых, в формулах используются числа, обозначения клетки и групп клеток, знаки арифметических действий и обращения к встроенным функциям. Числа записываются по обычным правилам (в "естественном" формате), только в качестве десятичной запятой используется точка. Знаки арифметических операций традиционны: + – ★ / , ^ – возведение в степень. Встроенные функции приведены в табл.3. В простейшем случае формула состоит всего лишь из обозначения одной клетки, на которую таким образом делается ссылка, или одного числа.

Параметры функций заключаются в круглые скобки. В некоторых системах ЭТ а именах встроенных функций дополнительно указывается спецсимвол – признак функции (часто это "@"). Тексты, числа и формулы пользователь вводит в той же строке экрана, что и команды. После нажатия клавиши "ввод" набранное автоматически переносится в так называемую активную, или текущую клетку.

Клетки обозначаются буквами и цифрами. Например, клетка в левом верхнем углу таблицы имевт наименование А1 (или а1, большие и маленькие буквы в обозначениях клеток Вариатаб-86 не различает), а клетка в третьей строке четвертой колонки – Г3. Группы соседних клеток можно обозначать сокращенно, например, Д2 : Д8.

Указатели колонок и рядов присутствуют на экране вверху и слева соответственно. Видимый при этом на экране размер клетки по умолчанию составляет 9 символов. Его легко изменить командой форматирования.

Тeблицa 3

Встроенные функции Вариτаб-86

Математические	Логические
АБС (з)	ЕС[ЛИ](а,б[, в])
ЕКСП (з)	И(а,б)
ККОР (з)	ИЛИ(а,б)
МАКС (с)	
МИН (с)	**Специальные**
МОД (з1,з2)	
ОКРУГЛ (з,места)	ПОДБОР (з,с) или ПБ (з,с)
РАН [ДОМ]	НД
СР[ЕДНЕЕ] (с)	ОШ[ИБКА]
СУМ (с)	ВЕРНО
СЧЕТ (с)	ЛОЖНО
ЦЕЛ (з)	ИТЕР
LN (з)	ДАТАЛИ (з) или ДЛИ (з)
LOG[10] (з)	ТЕКСТЛИ (з) или ТЛИ (з)
PI	НДЛИ (з)
SIN/COS/TAN (радианы)	ОШИБЛИ (з) или ОШЛИ (з)
ASIN/ACOS/ATAN (число)	ЧИСЛОЛИ (з) или ЧЛИ (з)
Календарные	
ДАТ[А] (мм,дд,[гг]гг)	М[ЕСЯЦ] (д)
Д[ЕНЬ] (д)	Г[ОД] (д)
СЕГОДНЯ	ДЕНЬНЕД (д)
НОМДНЯ (з)	ЮЛДАТ (д)

Примечание. *В таблице приняты обозначения:* з – *значение или координата клетки;* с – *список клеток;* д – *данные;* а,б,в – *выражения.*

Высота клетки всегда равна одной строке. Поскольку размер ЭТ намного больше площади экрана, то видна лишь часть ее. Двигая экран по ЭТ, можно увидеть любую часть таблицы.

Как можно видеть из рис.32, таблица состоит как бы из двух слоев – переднего и фонового. Последний обозначен штриховыми линиями. Содержимое клетки на схеме показано двумя строками, причем верхняя – это фоновый слой, а нижняя слегка сдвинутая влево, – передний. Пустой передний слой прозрачен. Когда в клетку заносится числовое значение, оно попадает в оба слоя, которые становятся неразличимыми. Текст также попадает в оба слоя и отличается в фоновом наличием слева двойной кавычки, а текстовое значение – скобками и кавычками с двух сторон. Если же в клетку заносится формула, то она занимает только фоновый слой. До проведения вычислений передний слой пуст и формула

виднв сквозь него. После вычислений в переднив слои всех клеток, содержвщих формулы, попадеют числа или сообщения об ошибках. Тогдв формулы перекрываются ими и ствновятся не видны. Впрочем, содержимое фонового слоя текущей клетки всвгда отображаются в отдвльной информационной строке, в нижней части экрвна. Можно задать и особый режим показа таблиц с формулами. Таким образом, обычно пользователь видит на экране фрагмент таблицы, наполненный текстами, числами и результатами вычислений по формулам.

Для работы с электронной таблицей имеется особая система комвнд. Признаком команды, отличающей ее от текста и формулы, является первый символ / (косая черта, или слэш). После него дается имя команды из одной буквы. Затем, если необходимо, вводят параметры (обозначения клеток, форматы их отображения и т.д.). Имеется специальная информационная команда, имя которой – знак вопроса "?" (косая черта в этом случае не требуется). Эта команда никак не влияет на таблицу, ее назначение – вывод на экран в зависимости от ситуации различных справок. Список команд Варитаб-86 привведен в табл. 4.

Таблица 4

Команды ЭТ Варитаб-86

Имя	Назначение
/Бланк	Удаляет или очищает содержимое клетки или диаграммы
/Вставка	Добавляет пустые ряды и колонки
/Графика	Отображает данные в виде диаграмм
/Двигать	Перемещает ряд или колонку на новое место
/Ж–реЖим	Меняет режим отображения и пересчете таблицы
/Исполнить	Принимает данные и команды из файла .ISP , т.е. запускает командный файл
/Копия	Дублирует диаграммы или содержимое и форматы отображения клеток
/Ликвидация	Удаляет таблицу из оперативной памяти
/Множить	Размножает содержимое частей колонок и рядов
/Новая ЭТ	Помещавт в ОЗУ всю или часть хранившейся таблицы
/Окно	Разделяет экран на окна
/Печать	Выводит содержимое ЭТ на принтер, экран или диск
/Редактировать	Предоставляет для редактирования содержимое клетки
/Сортировка	Сортирует ряды или колонки таблицы
/Титул	Фиксирует верхние ряды и (или) левые колонки на экране
/Удалить	Удаляет ряды, колонки из памяти или таблицу с диска

Окончание таблицы 4

/Формат	Устанавливает форматы отображения
/Хранить	Сохраняет текущую таблицу на диске
/Ц–конеЦ	Завершает исполнение программы Вaритаб-86
/Ш–разреШить	Снимает защиту с клеток
/Щ–заЩита	Защищает содержимое клеток от изменения
//	Дополнительные команды управления простой базой данных

Из табл.4 видно, что Варитаб-86 располагает определенными графическими возможностями, а именно, числовая информация может быть изображена в виде разного рода диаграмм (круговых, столбчатых, линейных и др.), широко применяемых в экономике. Кроме того, имеется возможность формирования базы данных из материалов многих таблиц, что очень важно в любой практической работе, в которой многократно используются одни и те же документы. Помимо перечисленных, имеются команда = (знак "равно" задает переход на указанную далее клетку), ! (восклицательный знак – пересчет таблицы по имеющимся в ней формулам) и еще несколько подобных. В процессе ввода элементов команды на экране возникают краткие подсказки, поясняющие их. Для получения более подробной справки по нужному параметру или ситуации необходимо нажать функциональную клавишу F1.

Для ввода команд отведена предпоследняя строка экрана, причем следующая команда должна подаваться после исполнения предыдущей.

В Варитабе-86 существует два курсора – активной клетки и командной строки. Одна (активная) клетка таблицы выделяется повышенной яркостью фона (или цветом) самой клетки, а также повышенной яркостью номера ряда и буквы колонки на краях экрана. Это и есть курсор текущей клетки. Его можно перемещать по таблице клавишами-стрелками и командой =. Кроме выделения яркостью обозначение активной клетки помещается под столбец нумерации рядов. Правее, в этой же строке, показывается содержимое фонового слоя текущей клетки.

Курсор командной строки предстаапяет собой небольшой мерцающий прямоугольник, расположенный непосредственно под вводимым символом. Номер позиции этого курсора отображается в начале командной строки и отделяется символом > ("больше"). Сразу после символа > вводятся команда, текст, число или формула. В текст могут включаться символы псевдографики и другие из кодовой твблицы ПЭВМ. Псевдографика позволявт заключить таблицу в рамку с графлением на отдвльные части, что улучшит наглядность и придаст эстетичный вид машинной распечатке.

При размножении содержимого клеток предусмотрена настройка формул. Это означает, что один раз введенную формулу, например, в верхнюю клетку колонки, можно скопировать в другие клетки, расположенные ниже, причем обозначения клеток, использованные в формуле, автоматически поменяются на правильные. Названный сервис является одной из важнейших черт всех систем электронных таблиц.

Пример. Пусть в таблице имеется 4 колонки, причем в первой содержится текст (например, наименование данных в рядах), во второй и третьей – независимые числа, а значения четвертой вычисляются как производное второй плюс третьей, умноженной на 0,15. Всего в таблице 31 ряд. В первых 30 рядах содержатся указанные данные, а в последней должны быть подведены итоги (т.е. вычислены суммы) по второй, третьей и четвертой колонкам. Изложенных выше сведений достаточно, чтобы понять, как формировать такую таблицу.

Сначала загружаем в ОЗУ Варитаб-86 запуском на исполнение файла W86.COM. Через некоторое время появляется начальная заставка, которую снимаем клавишей "ввод". На экране появляется пустая таблица.

```
    | А | Б | В | Г | Д | Е | Ж |
1
2
3
...
20
→ А1  (строка фона активной клетки)
      (строка подсказок по вводимым командам)
1 >   (строка команды и ввода)
F1 = Инф. F2 = Сброс ... (смысл функциональных клавиш)
```

Клетка А1 выделена в начальный момент повышенной яркостью, это активная (текущая) клетка. Наберем для нее в командной строке текст, начав его символом". Нажмем клавишу "ввод". Текст переместится в клетку А1, командная строка очистится, активной станвт клетка Б1. Чтобы удобнве было работать (меньше двигать курсор клетки), сначала будем заполнять первую колонку, затем другие. Вернем курсор клетки на А1, сдвинвм его на А2 стрелками "влево" и "вниз". Теперь направление автоматического смещения курсора будет "вниз". Заполним вторую и последующие клетки колонки А тем же приемом. Заметим, что теперь курсор клетки прыгает вниз, в не вправо. Заполнив колонку по клетку А30, дадим команду =Б1. Курсор клетки перескочит на самый верх колонки Б, и станет снова видна верхняя часть таблицы, а именно первые 20 рядов. Переместить курсор на Б1 можно и стрелками за 31 нажатие, и клавишами "Home", "стрелка вправо" – за два нажатия.

Набврем в строке ввода число для Б1, нажмем клавишу "ввод". Если курсор ушел не на Б2, вернем его туда, пройдя через клетку Б1. Теперь снова зафиксируется направление движения вниз, и можно вводить данные второй колонки, не задумываясь о курсоре клетки. В болве соввршенных системах ЭТ ввод данных в клетку можно заканчивать кла-

вишей-стрелкой требуемой ориентации, что заставит курсор переходить сразу на нужную соседнюю клетку, где бы она ни была. Заполняя колонку, заметим, что если текст прижимался к левому краю своей клетки, то числа прижимаются к правому. Этой особенностью ЭТ Вариатаб-86 также можно управлять (см. команду "формат"). Обратим внимание и на то, что когда мы подходим курсором клетки к нижнему краю видимого фрагмента таблицы, экран автоматически смещается по ней вниз.

Дойдя до Б30, снова выполним команду =В1 (или сместим курсор иным способом). Напомним, что команда исполняется только по нажатию клавиши "ввод". Заполним колонку В и перейдем на Г1. Нам нужно, чтобы в нее был занесен результат вычислений по формуле Б+0,15★В. Заносим в Г1 формулу б1+0.15★в1"ввод". Попутно напомним, что строчные и прописные буквы в формулах и командах воспринимаются одинаково, поэтому нет необходимости переключать лишний раз регистры.

Видим, что вместо формулы в Г1 появился сразу результат вычислений по ней (передний слой клетки с числом сделал невидимой формулу, которая находится на втором плане). Впрочем, переведя курсор клетки на Г1, мы тут же увидим находящуюся в ней формулу в четвертой снизу строке экрана. Теперь нужно сделать так, чтобы в остальных 29 клетках колонки Г появились аналогичные формулы. Это достигается размножением содержимого Г1 в клетки Г2, Г3,..., Г29. Введем команду "множить":

/М, Г1, Г2 : Г30 "ввод"

Заметим, что вместо клавиши "," (запятая) можно нажимать "ввод". Кроме того, запятая вслед за буквой имени команды появляется сама.

После выполнения команды в клетке Г1 появится формула Б2+0.15★В2, а клетке Г3 – формула Б3+0.15★В3 и т.д. А фактически на экране мы увидим сразу результаты счета по ним. Отметим попутно, что формулы правильно настраиваются при размножении как рядов, так и колонок.

Чтобы в 32-м ряду получить суммы по колонкам Б,В и Г, сделаем активной клетку Б31. Наберем в строке ввода обращение к функции суммирования: СУМ(Б1:Б30)"ввод". В клетке Б31 появится число, равное сумме всех чисел клеток Б1, Б2,...,Б30. Скопируем формулу из Б31 в клетки В31 и Г31. Правильные суммы вычислятся после настройки формул и появятся в переднем слое и в этих клетках.

Попробуем теперь поменять значение какой-нибудь клетки колонки Б, например, клетки Б7. Сделаем эту клетку активной, наберем в командной строке новое число и нажмем клавишу "ввод". Значение в клетке Б7 станет другим. Вместе с этим произойдут изменения еще в трех – Б31 (сумма по колонка), Г7 (поскольку Г7 зависит от Б7 по формуле) и Г31 (новая сумма по колонке Г из-за Г7). Как мы только что убедились, с электронной таблицей в отличие от обычной легко проводить эксперименты типа "что будет, если...". Такие эксперименты нужны, например, при конструировании изделия заданного веса, когда можно варьировать веса отдельных блоков. Причем надо отметить, что даже на таких мало-

мощных машинах, как IBM PC/XT, к которым относится "Правец-16", реакция ЭТ Вариатаб-86 в нашем примере будет практически мгновенной, хотя машине придется выполнить довольно много действий при пересчете формул. Длительность пересчета таблицы становится заметной (составляет секунды и минуты), когда размер таблицы достигает сотен рядов и колонок, наполненных более сложными формулами.

Календарные функции позволяют гибко манипулировать с датами: считывать текущую дату с таймера машины в клетку, задавать новую дату, выделять из даты номер дня, месяца и года, определять порядковый номер дня недели, вычислять номер дня с начала года. Группа специальных функций, имена которых оканчиваются на "-ЛИ" (рис.32), позволяют определять характер содержимого клеток. Это дает возможность автоматизировать учет характера вводимой информации и результатов предыдущих вычислений.

Остановимся немного на функции ПОДБОР. Она интересна тем, что имитирует табличную функцию, значения которой задает пользователь в виде пары рядом расположенных отрезков колонок (или рядов). При этом левая колонка (верхний ряд) должна содержать числовые значения аргумента, расположенные в возрастающем порядке, а правая колонка (нижний ряд) – числовые, текстовые или иные значения функции. Первым параметром функции ПОДБОР является число, для которого надо подобрать подходящее значение табличной функции по интервалу клеток аргументов, заданному вторым параметром. ПОДБОР находит в интервале клеток такую, число в которой не превышает заданного значения, и выдает содержимое соседней клетки справа (для колонок) или снизу (для рядов).

Вариатаб-86 позволяет создавать и исполнять командные файлы. Тем самым удается автоматизировать выполнение часто повторяющихся работ. Командный файл формируется как колонка в пустой таблице, наполненная текстами. В каждую клетку заносится цепочка символов, составляющая команду. Затем такая специфичная таблица по определенным правилам записывается на диск и в дальнейшем запускается на выполнение командой "Исполнить". Командный файл можно создать и любым текстовым редактором, поскольку это просто набор текстовых строк.

Более совершенные системы ЭТ, например, SuperCalc-4, имеют целый ряд дополнительных возможностей: расширенный набор математических, статистических, финансовых функций; механизм именования часто используемых блоков, упрощающий обращение к ним в различных командах; механизм вычисления координат клеток, предусматривающий использование координат текущей клетки; механизм создания макрокоманд, являющихся аналогами командных файлов, в той же самой таблице и их исполнения, причем а) внутри одной макрокоманды возможно исполнение других; б) макрокоманду можно обозначить однобуквенным именем, и тогда для ее запуска будет достаточно одновременного нажатия всего двух клавиш (Alt и символьной клавиши с именем макрокоманды).

Перечисленные усовершенствования позволяют чрезвычайно гибко манипулировать таблицами, а для часто повторяющихся работ очвнь легко создавать новые, спвциализированные команды. Поясним сказанное следующим примером. Допустим, что при работе с таблицей приходится часто "чистить" содержимое прямоугольного блока клеток, состоящего из отрезков двух столбцов и четырех рядов, т.е. всего из восьми клеток, причем такой блок встречается во многих местах таблицы. Чтобы "очистка" отнимала поменьше времени и усилий, имеет смысл подготовить для нее специальную макрокоманду. Сначала поименуем блок клеток A1:A2, в котором будет находиться команда чистки (/Blank в системе SuperCalc-4). Возьмем, например, имя \Z, руководствуясь тем, что клавиши Alt и Z на большинстве клавиатур расположены рядом.

/Name,\ Z, A1 : A2

Отметим, что макрокоманду можно поместить в любую часть электронной таблицы, где она не будет мешать формированию ведомости.

Среди встроенных функций имеются координатные – CCOL (Current Column – номер колонки текущей клетки) и CROW (номер ряда текущей клетки). Воспользуемся ими в команде чистки блока, которую введем как текстовую строку в клетку A1:

"/B [CCOL; CROW] : [CCOL + 1; CROW + 3] ~

Здесь больше подходит другой способ адресации, суть которого должна быть ясна из следующих прмеров:

[3; 5] эквиваленто C5
[7; 10] эквивалентно G10
[2; 6] : [4; 8] эквивалентно B6 : D8

Символ ~ (тильда) отмечает конец команды, т.е. заменяет клавишу Enter ("ввод").

Чтобы очистить любой блок клеток размером 4x2, достаточно установить курсор клетки в его левый верхний угол и нажать одновременно клавиши Alt и Z. Уже на таком элементарном примере видна польза от аппарата макрокоманд. Она гораздо ощутимее, если приходится иметь дело с длинными последовательностями команд.

Кратко коснемся графических возможностей электронных таблиц. Команда построения графика табличной функции требует выполнения как минимум трех входящих в нее подкоманд: а) выбор типа графика (круговая и столбчатая диаграммы, функция $y = f(x)$ и т.п.); б) задание блока клеток, из которого будут браться числовые значения; в) собственно построение графика в оперативной памяти и показ его на экране.

Кроме того, дополнительными подкомандами можно установить пределы изменений аргумента и функции, число шагов (отметок) по осям графика, заказать нанесение координатной сетки и надписей. Разные функции отмечаются на экране разным цввтом, штриховкой, значками, изображающими точки значений (квадратики, треугольнички и др.).

Подготовленный график можно сохранить на диске вместе с таблицей или вывести на бумагу. Кроме печати графика в ЭТ предусмотрены определенные возможности распечатки таблиц, что позволяет средствами ЭТ готовить полностью законченные документы (статьи, отчеты, справки), не требующие какой-либо ручной доработки.

Для передачи содержимого таблицы в другие системы (экспорт) и приема (ввода) информации в "чужих" форматах (импорт) предусмотрены специальные команды. Они облегчают задачу стыковки ЭТ с другими системами обработки информации, например, с текстовыми редакторами или системами управления базами данных.

Функции системы электронных таблиц SuperCalc-4 позволяют создавать и эксплуатировать базы данных. Правда, эти средства не такие богатые, как в мощных системах управления базами данных, но достаточные для решения многих практических задач.

Читатель, видимо, уже понял, что область использования ЭТ выходит далеко за рамки бухгалтерской сферы. С их помощью легко проводить всевозможные инженерные и научные расчеты, в том числе и весьма сложные, обрабатывать результаты опросов, наблюдений и измерений, решать системы уравнений и вычислять интегралы, получать наглядные графические изображения числовых данных и оформлять полученные материалы в виде печатных документов. Концепция ЭТ оказвлась на практике чрезвычайно продуктивной, и изобретательные пользователи нашли злектронным таблицам множество полезных применений.

ПРИМЕНЕНИЕ ЭВМ

Области применения ЭВМ сегодня очень разнообразны: научные, инженерные и экономические расчеты, моделирование процессов и явлений, проектирование и конструирование зданий, сооружений и изделий, выдача справочной информации по любому поводу и на любую тему, контроль и управление техническими устройствами и объектами, продажа товаров и услуг, изобразительное и музыкальнов творчество, обучение, игры, т.е. ЭВМ можно применять всюду, где человек имеет дело с информацией в любом ее виде. Обзор всех сфер реального применения компьютеров и имеющихся программных средств, разработанных в мире, может составить многотомный справочник.

Глввная тенденция развития товарного программного обеспечения, выпускаемого на продажу, состоит в увеличении объема функций, расширении сервиса для потребителя с одновременным упрощением и улучшением пользовательского интерфейса. Забота о расширении круга

потребителей за счет облегчения процедуры пользования ЭВМ является мощным стимулом для зарубежных производителей программных средств.

Харвктерный пример в этом плане – простые программные калькуляторы (ПК). При работе на ЭВМ с их богатыми возможностями остается потребность в эпизодическом выполнении элементарных расчетов. Программы в таких случаях писать невыгодно, быстрее получится, если просто посчитать "вручную". Здесь оказывается полезным ПК, который имитирует на ЭВМ обычный микрокалькулятор. Программный калькулятор может присутствовать в составе операционной системы как самостоятельная программа (например, CALC в Системе виртуальных машин ЕС ЭВМ) или быть встроенным в спвциализированную программную систему. Научиться пользоваться ПК можно буквально за считанные минуты.

Более удобными инструментами для выполнения всевозможных расчетов являются программные средства, называемыв интеллектуальными калькуляторами. Унаследовав присущую ПК простоту обращвния, они приобрели развитый математический аппарат, позволяющий решать сложные научные и инженерные задачи. Срвди известных зарубежных систем такого рода можно назвать TK!Solver, Eureka, MathCAD, Matlab, Gauss. Эти системы различаются составом математических функций и заложенных в них численных методов, способом организации диалога с пользователем, областями применения. Всем им присущи такие важные черты, как 1) наличие богатого набора готовых функций и простого гибкого инструмента их комбинирования; 2) существование встроенных средств редактирования вводной информации (заданий на решение): 3) возможность наглядного представлвния результатов вычислвния в виде таблиц, графиков и даже трехмврных изображвний; 4) наличие удобных диалоговых средств, позволяющих малоподготовленному пользователю общаться с системой без посторонней помощи; 5) устройчивость ко всевозможным ошибкам.

Применение подобной системы не требует знания языка программирования. Необходимо лишь определить, подходит ли она для решения аашего круга задач (достаточен ли набор встроенных функций, хороши ли средства управления системой, устраивает ли форма выдачи результатов и т.д.), и затем в течение непродолжительного врвмени освоить ве. Затраты времени на изучение подобной системы, ранее незнакомой, примерно равны затратам на программирование решения одной задачи средней сложности традиционным способом. Но зато с помощью хорошо освоенного инструмента такие задачи решаются гораздо быстрее. Например, большинство из названных систвм начинают выводить на экран график заданной функции немедленно (через доли секунды) после завершвния набора формулы (точнее, после нажатия клавиши "ввод"). Правда, вывод подробного графика несколькими сотнями точек можвт потребовать какого-то времени на их вычисление.

Запуск системы в работу осуществляется просто вызовом на исполнение головного файла из числа входящих в нее. После запуска на дисплей выдавтся начальная заставка, в которой указывается название, версия системы и ве авторы, а также минимум функциональных клавиш, применяемых для начала и завершения работы.

Если нынешние темпы создания подобных программных средств сохранятся, то можно прогнозировать через несколько лет резкое относительное, а в дальнейшем и абсолютное снижение потребности в традиционном программирование на языках высокого уровня при решении задач вычислительного характера. Понадобятся умения и навыки более высокого интеллектуального уровня. Мы здесь имеем в виду, что, например, работа по созданию экспертной системы – качественно иная деятельность, чем программирование на уровне элементарных арифметических операций и функций.

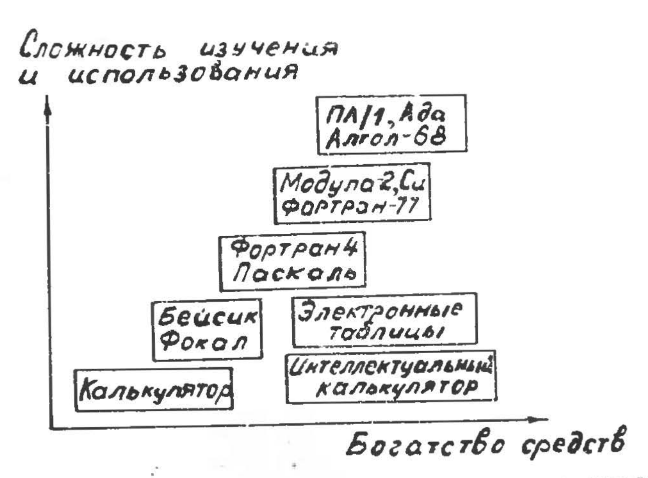


Рис. 33. Соотношение инструментальных программных срвдств

На рис.33 показано примерное соотношение между калькуляторами и языками высокого уровня в координатах "сложность использования – богатство возможностей". Приведены только языки с развитыми вычислительными средствами.

Для терпеливого читателя, желающего основательно изучить квкой-нибудь развитый современный язык прогрвммирования, можно рекомвндовать пройти 1–3 промежуточных этапа (языка), например, цепочки "Бейсик – Паскаль – Модула-2 – Ада"; "Паскаль – Алгол-68"; "Бейсик – Фортран-4 – Фортран-77", "Паскаль – Си". Освоение "промвжуточного" сравнительно простого языка поможет восприятию и пониманию конструкций следующего, более сложного.

Виды программных средств и пути их распространения

Чтобы сориентироваться в множестве готовых программных средств, необходимо их как-то разделить на классы и категории. Принципов классификации может быть несколько: 1) по выполняемым функциям, 2) языку программироеания, 3) типам (семействам) ЭВМ, 4) типам операционных систем, 5) фирмам-изготовителям ПС, 6) административно-территориальному признаку, 7) языку диалога (русский, английский,...) и т.д. Можно придумать и другие варианты, но на практике применяются е основном лишь назеанные.

В нашей стране централизованное распространение программных средств поручено Государственному фонду алгоритмов и программ (ГосФАП), состоящему из совокупности отраслевых и частично функциональных фондов алгоритмов и программ (ОФАП), а также республиканских фондов (РФАП).

Направляемые в ГосФАП программные средства оформляются в соответствии с требованиями ГОСТ ЕСПД – Единой системы программной документации. Минимальный комплект документов включает в себя "Описание программы" (т.е. ее внутреннее устройство). "Описание применения" (т.е. для чего она и как ею пользоваться). "Текст программы" (распечатка исходного текста на языке программирования или рабочая программа на машинном носителе). В комплект включается также информационная карта, в которой указываются основные функции программы и применяемый математический метод, язык программироеания, тип ЭВМ и операционной системы. Содержание информационных карт публикуется в ежемесячном информационном бюллетене ГосФАП "Алгоритмы и программы". В них программные средства группируются по функциям, а раньше классифицировались по типам ЭВМ и языкам. Бюллетень "Алгоритмы и программы" имеется во всех научно-технических библиотеках. Для получения копии документации по заинтересовавшему его ПС пользователь направляет в ГосФАП заявку по установленной форме, приведенной в конце бюллетеня. Некоторые отраслевые ФАП занимаются распространением ПС на машиночитаемых носителях (магнитных лентах, дискетах). В издаваемых этими ФАП каталогах даются описания ПС, цены на них и условия поставки.

В рубрикаторе ИПС ФАП (информационно-поисковой системы) принята тематическая классификация программных средств. В соответствии с ней размещаются описания ПС в бюллетенях ГосФАП и Государственной научно-тvхнической библиотеки, начиная с 1984 года. В рубрикаторе используется десятичная ступенчатая нумерация, в соответствии с которой основные рубрики имеют двузначные номера от 00 до 99, любая более мелкая рубрика получает дополнительный двузначный номер, записываемый через точку. В принципе может быть много ступеней, например:

06.75.71 Оплата труда. Заработная плата
20.23.21 Информационно-поисковые системы
50.05.13.03.07 Программирование "сверху вниз"
50.41.17.07.02.07 Интерпретаторы

Приведем небольшой фрагмент рубрикатора, занимающего на самом деле много страниц.

27 Математика	52 Горное дело
28 Кибернетика	55 Машиностроение
29 Физика	59 Приборостроение
31 Химия	78 Военное дело
39 География	82 Организация и управление
44 Энергетика	86 Охрана труда
49 Связь	90 Метрология

ГПНТБ СССР издает свой ежемесячный бюллетень "Алгоритмы и программы. Библиографическая информация", е котором сообщаются сведения о программах, полученные из всяких других источников, кроме ГосФАП. Бюллетень ГПНТБ дает много информации о зарубежных программных средствах, чего нет в бюллетенях ГосФАП. Как правило, источниками являются журналы и книги, поступившие в ГПНТБ. Никакой дополнительной информации по ПС, кроме самого первоисточника, библиотека не предоставляет.

Популярные зарубежные компьютерные журналы, как, например, BYTE, IEEE Software, PC World, публикуют много разнообразных сведений о программных средствах, включая цены и адреса разработчиков и распространителей (продавцов). Существуют фирмы оперативного тиражирования программ в режиме "горячей линии". Такие фирмы работают круглосуточно все дни недели и предоставляют свои услуги, т.е. передают копии программ, по телефону. Необходимо лишь, связавшись с фирмой, подключить на несколько минут свой персональный компьютер через модем к телефонной линии. Чтобы получить нужную программу в любой момент, владельцу ПЭВМ даже нет необходимости выходить из дома. Очевидно, что такая форма обслуживания обеспечивает практически мгновенное распространение новейших ПС и их версий с улучшенными характеристиками.

В нашей стране продажа программных средств изготовителем потребителю часто осуществляется по договорным ценам, хотя настоящий рынок такого товара еще предстоит создавать. Следует предостеречь читателя от покупки программной продукции у продацов, не имеющих законных оснований на продажу, проще говоря, торгующих краденым. К сожалению, вопрос охраны авторских прав разработчиков программ пока не получил в нашем законодательстве надежной правовой базы. Впрочем, и в зарубежных законодательствах в этом вопросе нет ни полной ясности, ни 100%-х гарантий, как нет и абсолютно надежных технических рвшений по защите от копирования.

Общение с ЭВМ на естественном языке

Очень важной областью крупномасштабного применения компьютеров является речевое или языковое общение. Под этим мы подразумеваем две основные проблемы: 1) восприятие с голоса и синтез человеческой речи; 2) машинный перевод с одного естественного языка на другой. Достигнутые в этих направлениях успехи еще не позволяют считать решенными все вопросы, а их оказалось достаточно много.

Что касается распознавания слитной человеческой речи, то до сих пор не удалось полностью расшифровать ее фонетический код. Поэтому не удалось создать устройства автоматического распознаавания речи даже в переделах одного языка. Построенныв системы либо имеют очень ограниченный словарь узнаваемых слов, либо требуют настройки на голос и произношение определенного человека.

С синтезом речи квртина аналогичная. Элементарная разборчивая речь из двух-трех десятков отдельно произносимых слов реализована давно. За рубежом уже выпускаются микрокалькуляторы для слепых, в которых нажатие на любую клавишу вызывает произнесение ее названия либо получившегося результата. Более совершенный вариант, встроенный в ЭВМ, дает побуквенное и слитное произношение вводимого или содержащегося а памяти текста. Несмотря на очевидную ограниченность полученного результата, открываются большие аозможности доступа к информации для слепых людей через компьютеры, особенно компьютерные сети. Персональная ЭВМ, оснащенная текстовым сканером и синтезатором речи, может служить незаменимым информационным помощником для слепых. Им станут доступны без посторонней помощи обычные книги, журналы, газеты. С несовершенством синтезироаанной речи в этом случае вполне можно смириться.

Простые синтезаторы уже встраивают в твлефонные системы, предназначенные, например, для оповещения клиентов о необходимости внесения абонентской платы. В целом же синтезированная речь очень монотонна, обладает неприятным металлическим тембром, не всегда разборчива. Предстоит большая работа по ее улучшению.

Интересный и забавный пример устройствв, понимающего и воспроизводящего человеческую речь, – кукла Джулия, выпускаемая в США. Она распознавт в задаваемых ей вопросах ключевыв слова и звтем достаточно правильно отвечает фразами, записанными в синтеэаторв речи, расположенном а кукле. При изменении температуры воздуха или положения куклы, определяемых встроенными датчиками, Джулия может сквзать, что ей холодно или жарко, и даже спросить, куда ее ведут. Общенив с куклой, разумеется, идет на английском языке.

В настоящее время за рубежом разрабатывается ряд проектов, связанных с распознаванием и синтезом слитной речи. Это информационно-справочные системы различного назначения, в которых абонент устно запрашивает интересующие его сведения и получает ответ в устной

или письменной форме. Американские и японские фирмы работают над системой, обеспечивающей устный ввод и печатание слитно произнесенного текста. По сравнению с дискретным вводом слов в существующих уже печатных машинках использование слитной речи повышает производительность печатания в 2-3 раза. Предполагаемый срок создания экспериментальной пишущей машинки – 1995 год. Основа ее конструкции – все тот же компьютер.

Проблема машинного перевода текстов с одного естественного языка на другой давно интересует ученых многих стран. Прорабатывается несколько схем перевода. Одна из наиболее удачных выглядит так: текст входной → ввод в ЭВМ → переводческая система → текст промежуточный машинный → редактор → текст выходной.

В этой схеме, по существу, две фазы перевода. На первой работает переводческая система, которая порождает эквивалентный входному промежуточный текст. На второй фазе "сырой", грубо сделанный машинный перевод правит человек-редактор. Использование этой технологии в разных переводческих бюро показало повышение производительности труда в 2-3 раза, а в некоторых особых случаях и до 5 раз.

В большинстве систем перевод осуществляется отдельными фразами. Связи, выходящие за пределы фразы, анализируются очень редко. Полностью автоматизированный грамматически правильный перевод пока удается только для сообщений с весьма ограниченной лексикой. Например, без участия человека переводятся метеосводки. Есть надежда добиться также полной автоматизации перевода научно-технических текстов. Перевод художественной литературы посредством ЭВМ сегодня не предполагается даже в принципе.

Наибольшее распространение в мире среди переводческих систем получили версии системы SYSTRAN, в которой уже реализованы 15 пар языков: с английского – на голландский, испанский, итальянский, немецкий, португальский, русский, французский, японский; с французского – на английский, голландский, немецкий; с испанского, немецкого, русского и японского – на английский.

Интересные перспективы открывает идея соединения в одну систему средств устного ввода-вывода и машинного перевода. Можно представить себе разговор по телефону двух лиц, говорящих на разных языках, невидимым посредником которых был бы такой электронный переводчик. В принципе его создание для разговорной речи с ограниченной лексикой уже возможно, дело за тем, чтобы втиснуть это устройство в приемлемые габаритные размеры и уложиться в небольшую стоимость. А как было бы полезно для нашей многонациональной страны иметь множестао таких встроенных в государственную телефонную сеть электронных переводчиков, обеспечивающих возможность общения друг с другом миллионов людей, говорящих на разных языках!

Электронный переводчик был бы и существенным подспорьем в изучении языков. Его можно было бы использовать для перевода де-

ловых сообщений, потребность в которых стремительно рвстет, и многого другого. На этой же основе могла бы быть изготовлена пишущая машинка, которая воспринимает с голоса текст на одном языке, а печатает его сразу на другом.

В заключение несколько слов о сетях ЭВМ. Эффективное использование мини-, микро- и в особенности персональных ЭВМ сегодня немыслимо без интенсивного обмена программами и данными. Перенос информации посредством магнитных носителей (дискет) удобен для территориально близко расположенных пользователей (в пределах одной организации, одного здания). Оперативная передача данных между удаленными ЭВМ, находящимися в разных зданиях, городах или странах, возможна по компьютерным сетям. В мире разработаны и эксплуатируются уже большое количество сетей, различающихся пропускной способностью (числом байт в секунду), предельным удалением получателя от источника информации, стоимостью пересылки единицы информации или единицы времени работы сети и т.д.

В зарубежных компьютерных сетях большое распространение получила связь посредством специальных устройств – модемов (модуляторов-демодуляторов), подключаемых к существующим телефонным линиям. Выпускаются модемы различных типов, работающих как через электрический разъем с телефонной линией, так и через акустический контакт с трубкой телефонного аппарата. В последнем случае в телефонной сети не делается вообще каких-либо соединений-переключений. Достаточно популярное и полное описание существующих компьютерных сетей читатель может найти, например, в [10].

Модем преобразует цифровые сигналы, поступающие с компьютера и выраженные уровнем потенциала, в частотно-модулированные сигналы, которые легко передавать по телефонным линиям, он также осуществляет прием модулированных сигналов, генерированных другим модемом, и преобразование (демодуляцию) их в цифровые сигналы, которые могут восприниматься и обрабатываться ЭВМ.

Модемы изготавливаются как в виде встроенной конструкции на сменной плате, так и в виде самостоятельного внешнего блока. Основу модема составляют его кристалл (большая специализированная интегральная микросхема) и развязывающий трансформатор. Скорости передачи колеблются от 300 и ниже до 1200 бод, а в высокоскоростных – до 2400, 4800 и даже до 9600 бод.

Кроме самого модема для осуществления связи требуется особая программа, соответствующая его конструкции, принятому протоколу передачи информации и, разумеется, операционной системе компьютера. Подробное и доступное описание существующих компьютерных сетей, протоколов, связной аппаратуры и программ можно найти в [5,10].

Что дальше?

Сегодня в мире насчитывается около 30 млн персональных ЭВМ, и каждый месяц около 1 млн вводится в действие. К концу 1990 года будет установлено около 40 млн ПЭВМ. Более 20 млн ПЭВМ работают под управлением операционной системы MS DOS и аналогичных ей. В последнее время основное развитие идет по пути наращивания параметров традиционных устройств – увеличения тактовой частоты процессора, расширения максимального объема оперативной памяти примерно до 10 Мбайт, повышения емкости дисковых накопителей, снижения габаритных размеров и цены лазерных принтеров. Вместе с тем появляются и такие модификации, которые позволяют говорить о принципиально новых устройствах. Назовем некоторые из них.

Новым устройством управления курсором, показанным впервые менее 2 лет назад, является манипулятор Isopoint Control. Он способен заменить "мышь" и "шар" в портативных ЭВМ и тех случаях, когда пользователь вынужден работать в условиях ограниченного пространства. Манипулятор представляет собой тонкий цилиндр, вращая который, можно передвигать курсор в двух направлениях. Нажатие на цилиндр аналогично нажатию кнопки на "мыши". С помощью этого манипулятора пользователь управляет курсором, не снимая рук с блока клавиатуры, поскольку устройство располагается на краю блока, ниже клавиши пробела. Это уменьшает физическую нагрузку на руки, заметную при интенсивном использовании клавиатуры и "мыши" одновременно. Кроме того, при наличии такого манипулятора можно отказаться от четырех клавиш-стрелок двиения курсора, что немаловажно в портативных ПЭВМ.

Перспективная модификация струйного принтера JUKI 8000 предложена японскими изготовителями. В нем в качестве красителей использованы черный, желтый, красный и синий пластиковые карандаши, которые плавятся при нагревании. Жидкие красители разбрызгиваются пьезоэлектрическими элементами. Капельки мгновенно затвердевают на бумаге и, будучи точно позиционированы, дают при наложении до 260000 оттенков цветов. Разрешение составляет 9,5 точек/мм на бумаге формата А4 (210x297 мм), время печати одной страницы – 2,6 мин. Собственное ОЗУ имеет емкость 512 Кбайт. Принтер выполнен в виде напольной конструкции весом 34 кг. В принтере устранен характерный для устройств этого типа недостаток – засорение струйных сопел высыхающей краской.

Интересную разработку накопителей на сменных дисках представили компании Proteus Technology и Sysgen. В семейство входят пять накопителей типа "винчестер" со сменными дисками емкостью от 20 до 150 Мбайт с высокой скоростью доступа. Диски имеют стандартный размер 133 мм, отличаются от гибких более высокой надежностью и стоимостью.

Совершенствование отдельных компонентов и устройств по всем параметрам должно привести к появлению в скором времени персональных компьютеров качественно нового уровня. Возможно, одной из таких революционных разработок, предназначенных специально для целей обучения, станет семейство компьютеров NeXT фирмы NeXT, Inc., являющееся развитием широко известной линии ПЭВМ Apple-Macintosh.

Основу машины NeXT составляет системный блок в виде куба с ребром около 30 см. В нем расположены универсальный блок питания, автоматически настраивающийся на любой национальный стандарт электросети, ОЗУ, емкостью не менее одного мегабайта, память дисплея, процессоры, накопитель на сменном оптическом диске диаметром 133 мм и емкостью 250 Мбайт, устройства для ввода с микрофона, звукового стереовывода, включения машины в сеть типа Ethernet, подключения манипулятора "мышь" и других внешних устройств, гнезда расширения. Монохромный черно-белый дисплей высокого разрешения содержит 1120x832 точек (пикселов), около 4 точек/мм. Дисплей такого качества позволит проводить больше времени за компьютером без риска потерять зрение. Блок клавиатуры с 84 клавишами подключается к дисплею. "Мышь" с двумя кнопками подсоединяется к клавиатуре. Куб может удаляться от дисплея на расстояние до 3 м, его удобно ставить на полку, освобождая место на столе. Малогабаритный лазерный принтер LaserWriter дает высококачественные распечатки с текстовым и графическим содержанием.

Особое внимание разработчики уделили звуковым средствам, в результате чего NeXT может слышать, говорить, воспроизводить и синтезировать мелодии с очень высоким качеством звучания. Математический сопроцессор с тактовой частотой 25 МГц обеспечивает проведение сложных расчетов, а цифровой процессор сигналов – распознавание речи и синтез всевозможных звуков.

Программное обеспечение впитало в себя лучшее из того, что накоплено для ПЭВМ Macintosh, и новые разработки как в области инструментария для программиста, так и объектно ориентированные комплексы, в частности, для синтеза музыкальных произведений, восприятия и генерации человеческой речи с обширным словарем и классическим английским произношением. Объявленная цена машины составляет для учебных заведений 6500 американских долларов. Более подробную информацию читатель найдет в сборнике "Мир ПК", № 5 за 1989 г. и в [20].

* * *

Автор надеется, что приведенные в книге сведения и личный опыт помогут начинающему пользователю преодолеть робость, непонимание и чувство неуверенности, знакомое многим взрослым по их первым контактам с ЭВМ. Терпеливый читатель сможет быстро перешагнуть невидимый, но вполне ощутимый "барьер вхождения", лежащий на пути в компьютерный мир. Стоит затратить немного усилий, проявить настойчивость, и современный компьютер, скорее всего это будет персональная ЭВМ, доставит вам много интересных минут, принесет ощутимые результаты и удовлетворение, какие может дать общение с очень толковым и исполнитвльным помощником.

ТЕРМИНОЛОГИЧЕСКИЙ СЛОВАРЬ *

АВТОКОД – машинно-ориентированный язык программирования, позволяющий использовать все особенности системы команд процессора ЭВМ. При этом программирование осуществляется в терминах мнемонических обозначений машинных команд (операций) и их операндов.

АВТОМАТИЗИРОВАННАЯ ОБРАБОТКА ДАННЫХ – вычислительная система для обработки данных, характеризующих объект или явление, она формирует результаты обработки в виде, удобном для хранения и анализа.

АВТОМАТИЗИРОВАННАЯ СИСТЕМА ОБУЧЕНИЯ – комплекс аппаратных и программных средств, позволяющий обучать и контролировать знания.

АДА – язык программирования высокого уровня для применения в системах реального времени; по замыслу его авторов, обеспечивает надежность программ и удобство их разработки; создан в начале 80-х годов.

АДАПТЕР – устройство, осуществляющее необходимое преобразование информации для согласования каналов (устройств) разных типов. В ЭВМ адаптеры обеспечивают обмен между ОЗУ и ВЗУ, ОЗУ и устройствами ввода-вывода.

АДРЕС – цифровое или буквенно-цифровое обозначение местонахождения элемента информации в памяти ЭВМ (например, номер ячейки или машинного слова).

АЛГОЛ [от algo (rithmic) l (anguage)] – универсальный язык программирования высокого уровня, применяемый для формализованной записи алгоритмов, создан в 60-е годы.

АЛГОЛ-68 – поздний и значительно усиленный вариант языка Алгол; развитие и совершенствование его продолжается по настоящее время.

АЛГОРИТМ – сформулированная на некотором языке последовательность действий, строгое выполнение которой дает точное решение задачи. Алгоритм характеризуется дискретностью, массовостью, определенностью, результативностью и формальностью.

АЛГОРИТМА БЛОК-СХЕМА – см. БЛОК-СХЕМА.

---

* При подготовке словаря автор использовал следующие первоисточники: Терминологический словарь по автоматике, информатике и вычислительной технике: Справочное пособие для СПТУ/В.В.Зотов, Ю.Н. Маслов, А.Е. Порядочкин и др. – М.: Высшая школа, 1989. – 191 с.; **Заморин А.П., Мячев А.А., Селиванов Ю.П.** Вычислительные машины, системы, комплексы: Справочник. – М.: Энегоатомиздат, 1985. – 264 с.

АЛГОРИТМА ДИСКРЕТНОСТЬ – свойство алгоритма, означающве, что процесс првобразований, описываемый алгоритмом, должен иметь дискретный (прерывистый) характер, т.е. двлиться на отдельные законченные шаги.

АЛГОРИТМА МАССОВОСТЬ – применимость алгоритма ко всем задачам рассматриваемого типа при любых исходных данных.

АЛГОРИТМА ОПРЕДЕЛЕННОСТЬ – свойство алгоритма, состоящве в том, что каждое входящее в него действие должно быть строго определено и не допускать различных толкований. Так же строго должен быть определен и порядок (последовательность) выполнения отдельных действий.

АЛГОРИТМА РЕЗУЛЬТАТИВНОСТЬ – алгоритм всегда должен завершаться за конечное (возможно, очень большое) число шагов.

АЛГОРИТМА ФОРМАЛЬНОСТЬ – свойство алгоритма, позволяющее любому исполнителю (человеку, техническому устройству, программе, ЭВМ), способному воспринимать и выполнять указания алгоритма, правильно исполнить весь алгоритм. При обучении алгоритмическому мышлению часто применяют специально придуманных исполнителей ("муравей", "черепашка" и т.п.), способных формально, не вдумываясь в смысл, отрабатывать составленные алгоритмы.

АЛФАВИТ – конечное множество (обычно в пределах сотни) разных символов (букв, цифр, скобок, знаков препинания и других), используемых в данном языке программирования.

АРХИТЕКТУРА в информатике – концепция состава и взаимосвязи элементов сложной системы (вычислительной машины, программно-аппаратного комплекса, вычислительной сети и т.д.).

АССЕМБЛЕР – 1) машинно-ориентированный язык программирования типа автокода; 2) программа-транслятор для перевода с языка ассемблера на машинный язык.

АЦПУ – алфавитно-цифровое печатающее устройство – прежнве название устройства отображения результатов работы ЭВМ на бумаге, обычно имеющего барабанный механизм печати, значительные габаритные размеры и вес. АЦПУ до сих пор применяются в больших и средних моделях ЭВМ.

БАЗА ДАННЫХ (БД) – совокупность взаимосвязанных данных при предельно малой избыточности, допускающей их оптимальное использование в определенных областях человеческой деятельности. Базы данных в зависимости от способа представления данных и отношений между ними могут иметь реляционную, сетевую или иерархическую

структуры. На эффективность БД с той или иной структурой влияют условия ее примвнения. Программное обеспечение для управления и поддержки работоспособности БД называют системой управления базой данных (СУБД).

БАЗА ЗНАНИЙ – совокупность базы данных и интеллектуальных программ их преобразования в соответствии с правилами некоторой предметной области.

БАЙТ – единица измерения количества информации, равная обычно 8 битам, которой ЭВМ может манипулировать как единым целым. В байтах часто выражают объем программ и данных, а также емкость памяти ЭВМ.

БАНК ДАННЫХ – совокупность программных, языковых и технических средств, предназначенных для централизованного сбора, хранения и коллективного использования данных об определенной области человеческой деятельности, а также сами данные в виде файлов, баз данных и др.

БЕЙСИК (БЭЙСИК) – популярный язык программирования с относительно простым синтаксисом. Имеется на большинстве типов микроЭВМ. Обычно используется в режимв интерпретации для решения разнообразных задач.

БЕНЧМАРК [benchmark] – программа для сравнительной или абсолютной оценки быстродействия ЭВМ и систем программирования. Содержит специально подобранную "смесь" операторов и языковых конструкций, адекватную реальной или желаемой частоте их употребления в различных областях применения ЭВМ.

БИТ – минимальная порция информации, содержащаяся в сообщении типа "да-нет". В двоичном коде это один разряд машинного слова или байта, могущий принимать только два значения: 0 или 1 ("да" или "нет").

БЛОК – 1) составная часть программы ЭВМ, ограниченная специальным символом начала и конца блока. Существенным свойством блока является то, что оперативная память для описанных в блоке величин выделяется только в момент входа в блок при активизации описаний. Это позволяет одни и те же участки памяти использовать для хранения величин, определенных в разных непересекающихся блоках. Кроме того, переменным из разных блоков можно давать одинаковые имена (идентификаторы). Конструкция "блок" используется в ряде языков программирования высокого уровня; 2) обособленная часть ЭВМ, представляющая собой совокупность функционально объединенных элементов (например, блок питания).

БЛОК-СХЕМА – в программировании изображение структуры программы в виде геометрических фигур со стрелками-связями, показывающими последовательность работы отдельных частей.

БОД – единица измерения скорости побитной передачи информации: 1 бод = 1 бит/с.

БЫСТРОДЕЙСТВИЕ – один из основных параметров любой ЭВМ. Под быстродействием понимают некоторое усредненное число машинных команд, в том числе арифметических операций, выполняемых процессором в единицу времени (одну секунду). По уровню быстродействия и другим параметрам ЭВМ делятся на классы.

ВВОД РЕЧЕВОЙ – ввод информации в ЭВМ словами и фразами непосредственно с голоса с помощью микрофона и специальных программ распознавания речи.

ВЕРСИЯ – вариант постоянно совершенствуемой программы или операционной системы.

ВИДЕОТЕРМИНАЛ – см. ДИСПЛЕЙ.

ВИДЕОТЕРМИНАЛ ИНТЕЛЛЕКТУАЛЬНЫЙ – видеотерминал, содержащий собственные процессор и память, способные частично заменить центральный процессор.

ВИНЧЕСТЕРСКИЙ НАКОПИТЕЛЬ – запоминающее устройство большой емкости на жестких магнитных дисках с облегченными головками. Диски с головками и частью механизма позиционирования герметизированы, благодаря чему обеспечиваются высокая плотность записи, малое время доступа и высокая надежность. Современные "винчестеры" имеют емкость в десятки и сотни мегабайт. Широко применяются в составе персональных ЭВМ.

ВИРТУАЛЬНЫЙ – кажущийся, при помощи специальных средств моделируемый на реальном оборудовании и предоставленный в распоряжение, например, виртуальная ЭВМ, виртуальная оперативная память.

ВИРТУАЛЬНЫЙ ДИСК – реализуемый программно в оперативной памяти накопитель, обращение к которому производится как к реальному дисковому ЗУ. Отличается повышенным быстродействием, небольшой емкостью и свойством полной потери информации при отключении питания.

ВИРУС КОМПЬЮТЕРНЫЙ – небольшая (0,5–3 Кбайт), сложная, тщательно составленная и опасная программа, которая может самостоятельно размножаться, переносить себя на диски и дискеты, прикрепляться к чужим программам и передаваться по сети. Обычно создается для нарушения работы компьютеров различными способами – от "безобидной" выдачи какого-либо сообщения до стирания файлов и физической порчи жесткого диска. Выявление "вирусов" и "лечение" инфицированных файлов осуществляется разными методами, в том числе специальными антивирусными программами.

ВНЕШНЕЕ ЗАПОМИНАЮЩЕЕ УСТРОЙСТВО (ВЗУ) – см. ЗАПОМИНАЮЩЕЕ УСТРОЙСТВО ВНЕШНЕЕ.

ВЫВОД РЕЧЕВОЙ – вывод информации из ЭВМ в виде звуков человеческой речи, имитируемых с помощью программ и электронных устройств – синтезаторов.

ВЫЧИСЛЕНИЯ С ДВОЙНОЙ ТОЧНОСТЬЮ – выполнение арифметических операций над числами с удвоенным числом разрядов. Как правило, быстродействие ЭВМ на таких операциях в несколько раз ниже, чем при вычислениях с обычной точностью.

ВЫЧИСЛЕНИЯ С ПЛАВАЮЩЕЙ ТОЧКОЙ (ЗАПЯТОЙ) – "плавающая" арифметика – выполнение арифметических операций над числами, представленными в форме с плавающей точкой, т.е. в нормализованном внутреннем представлении, при этом в процессе выполнения операции автоматически меняется масштаб числа так, что количество значащих цифр остается постоянным. Тем самым обеспечивается одинаковая точность в широком диапазоне значений. В простых дешевых процессорах (например, в основных микропроцессорах большинства ПЭВМ) "плавающая" арифметика отсутствует. В этом случае она программно моделируется на основе других операций, что существенно сказывается на быстродействии при решении задач вычислительного характера, либо используется дополнительный, так называемый математический сопроцессор.

ВЫЧИСЛЕНИЯ С ФИКСИРОВАННОЙ ТОЧКОЙ (ЗАПЯТОЙ) – выполнение арифметических операций над числами с фиксированной точкой (запятой), когда положение точки, отделяющей дробную часть от целой, остается постоянным. Отличается повышенным по сравнению с "плавающей" арифметикой быстродействием.

ГЕНЕРАТОР ПРОГРАММ – специальная программа, предназначенная для формирования необходимой пользователю модификации программы определенного класса.

ГЕНЕРАТОР СЛУЧАЙНЫХ ЧИСЕЛ – нестрогое название небольшой подпрограммы (процедуры), дающей на каждое обращение очередное

ГИГАБАЙТ [Гбайт, GB] – объем памяти в 2^{30} = 1 073 741 824 байт, или 1024 Мбайт.

ГОЛОВКА МАГНИТНАЯ – устройство для записи информации на магнитных носителях. Представляет собой магнитопровод с зазором и расположенными на нем управляющими обмотками. Габариты собственно головки обычно не превышают нескольких мм.

ГРАФИКА МАШИННАЯ – воспроизведение выводимой из ЭВМ информации в виде графиков, чертежей и реалистичных изображений на бу-

маге или экране графического дисплея, получаемых на основе различных физических принципов. К машинной графике относят также программные средства построения изображений.

ДАННЫЕ – представление всевозможной информации в виде, позволяющем автоматизировать ее сбор, хранение и обработку в ЭВМ. Данные могут выражаться набором знаков, цифр, текстов, изображений. В частности, в квчестве данных могут выступать и тексты программ (по отношению к транслятору, например). Различают данные входные и выходные, но с развитием вычислительной техники эта разница постепенно исчезает: одни и те же данные могут быть входными а одном случае и выходными – в другом.

ДАННЫХ ПЕРЕДАЧА – область электросвязи, посвященная передаче информации, представленной в формализованном виде и предназначенной для ее обработки с помощью ЭВМ. Передачу осуществляют по телеграфным, телефонным или специально созданным для этого каналам связи. Вместе с компьютерной техникой каналы передачи данных являются аппаратной основой информационных, управляющих и вычислительных систем.

ДАННЫХ РЕДАКТИРОВАНИЕ – преобразование содержания данных, формы их представления и расположения к виду, удобному для использования, например, редактирование текста документа.

ДАННЫХ ТИП – совокупность свойств, характерных для некоторого набора данных, определяющая множество значений, которые могут принимвть эти данные.

ДАТЧИК СЛУЧАЙНЫХ ЧИСЕЛ – см. ГЕНЕРАТОР СЛУЧАЙНЫХ ЧИСЕЛ.

ДЕМОДУЛЯЦИЯ – извлечение из принятого модулированного сигнала модулирующего сигнала, представляющего собой информацию.

ДИАЛОГОВЫЙ РЕЖИМ – режим общения пользователя с ЭВМ, при котором каждый запрос пользователя вызывает немедленную ответную реакцию машины. В диалоговом режиме обычно работают текстовые и графические редакторы, инструментальные, игровые и многие другие программы.

ДИГИТАЙЗЕР (КОДИРУЮЩИЙ ПЛАНШЕТ) – устройство ввода графических данных векторного типа (чертеж, схема, план).

ДИСК МАГНИТНЫЙ – тонкий диск из немагнитного материала, покрытый с одной или двух сторон слоем ферромагнитного вещества. Информация записывается (считывается) магитными головками на концентрических дорожках при его вращении. Поиск нужной дорожки осуществляется смещением головки по радиусу диска.

ДИСКЕТА (ДИСКЕТ, ФЛОППИ-ДИСК) – носитель информации в виде гибкого магнитного диска, заключенного в неразборный гибкий или полужесткий пластиковый корпус. Очень широко используется в ПЭВМ в качестве дешевого сменного носителя приемлемой емкости.

ДИСКОВОД – разговорное наименование накопителя на сменном магнитном диске (без носителя информации).

ДИСПЛЕЙ – устройство визуального отображений информации. Выполняется в основном на электронно-лучевых трубках (кинескопах), жидкокристаллических, газоразрядных или плазменных панелях. Дисплеи ПЭВМ обычно совмещают алфавитно-цифровой и графический режимы.

ДИСПЛЕЙ АЛФАВИТНО-ЦИФРОВОЙ – дисплей для отображения только алфавитно-цифровой (текстовой) информации.

ДИСПЛЕЙ ГРАФИЧЕСКИЙ – дисплей для вывода графиков, рисунков и других изображений наряду с текстами.

ДОСТУП – процедура установления связи с запоминающим устройством ЭВМ для чтения или записи порции данных.

ДРАЙВЕР – 1) программа, реализующая обмен данными между основной и внешней памятью или устройствами ввода-вывода; 2) простая технологическая программа, используемая лишь для отладки и тестироввния подпрограммы (процедуры).

ДЖОЙСТИК – устройство для ручного управления движением курсора на экране дисплея. Выполняется в виде ручки или стержня на подставке и нескольких кнопок. Отклонение ручки от вертикали приводит к смещению курсора в соответствующем направлении. См. твкже МАНИПУЛЯТОР "МЫШЬ".

ЕДИНАЯ СИСТЕМА ЭВМ (ЕС ЭВМ) – семейство программно-совместимых ЭВМ третьего поколения, объединяющее ряд моделей единой архитектуры с производительностью от нескольких тысяч до нескольких миллионов операций в секунду. Обширная номенклатура периферийных устройств и стандартный интерфейс позволяют создавать вычислительные системы различной конфигурации. Богатое программное обеспечение в сочетании с разнообразными техническими средствами создает условия для широкого круга научно-технических, экономических, управленческих и других задач. В настоящее время выпускаются ЭВМ третьей очереди ЕС 1036, ЕС 1046, ЕС 1066 с улучшенными технико-экономическими и эксплуатационными характеристиками. Разработку и серийное производство программных и технических средств ЕС ЭВМ осуществляют Болгария, Венгрия, ГДР, Польша, Чехословакия, Румыния и СССР.

ЕМКОСТЬ ЗАПОМИНАЮЩЕГО УСТРОЙСТВА – наибольшее количество информации, которое можно одновременно хранить в запоминающем устройстве. Чаще всего измеряется числом байт и производных единиц (Килобайт – К, Мегабайт – М).

ЗАГРУЗЧИК – обслуживающая программа для ввода (загрузки) рабочей программы в оперативную память ЭВМ (как правило, с магнитного диска). В функции некоторых загрузчиков входит предварительная сборка программы из объектных модулей.

ЗАПИСЬ в операционной системе – одна из основных форм представления информации при ее вводе-выводе. Чаще всего одной записи соответствует одна строка текста.

ЗАПОМИНАЮЩЕЕ УСТРОЙСТВО (ЗУ) – устройство для записи, хранения и выдачи по запросу информации в ЭВМ и системах обработки данных. Состоит из накопителя (собстаенно запоминающей среды и привода, обеспечивающего доступ к отдельным ее участкам), блоков приема данных, записи, выборки, считывания, выдачи и устройства местного управления. К ЗУ относятся оперативное запоминающее устройство (ОЗУ – основная, или оперативная память ЭВМ), внешние запоминающие устройства (ВЗУ – накопители на магнитных дисках и лентах) и др.

ЗАПОМИНАЮЩЕЕ УСТРОЙСТВО ПОСТОЯННОЕ (ПЗУ) – ЗУ с неизменяемым в процессе эксплуатации информационным содержимым. Способно сохранять данные при длительном отключении питания.

ЗАПОМИНАЮЩЕЕ УСТРОЙСТВО ПОСТОЯННОЕ ПРОГРАММИРУЕМОЕ (ППЗУ) – ЗУ, в которое информация заносится самим пользователем однократно в начале эксплуатации и после этого не меняется.

ЗАПОМИНАЮЩЕЕ УСТРОЙСТВО, ПОСТОЯННОЕ ПЕРЕПРОГРАМИРУЕМОЕ – ЗУ, информация в которое заносится многократно пользователем, но время записи значительно превышает время считывания. Запись осуществляется специальными средствами. Сохраняет данные при отключенном питании.

ЗАПОМИНАЮЩЕЕ УСТРОЙСТВО СВЕРХОПЕРАТИВНОЕ (СВЕРХОПЕРАТИВНАЯ ПАМЯТЬ, СОЗУ) – быстродействующее ЗУ, применяемое в качестве регистров процессора для временного хранения часто используемых данных и констант. Емкость СОЗУ во много раз меньше емкости ОЗУ.

ЗАПОМИНАЮЩЕЕ УСТРОЙСТВО С ПОСЛЕДОВАТЕЛЬНЫМ ДОСТУПОМ – ЗУ, в котором доступ к участкам запоминающей среды осуществляется последоаательно один за другим. Типичный представитель – накопитель на магнитной ленте.

ЗАПОМИНАЮЩЕЕ УСТРОЙСТВО С ПРОИЗВОЛЬНЫМ ДОСТУПОМ – ЗУ, в котором возможен немедленный доступ к любой ячейке памяти. Пример – ОЗУ.

ЗАПОМИНАЮЩЕЕ УСТРОЙСТВО С ПРЯМЫМ ДОСТУПОМ – ЗУ, в котором возможен быстрый доступ к любому участку запоминающей среды. Типичный представитель – накопитель на магнитной ленте.

ЗАЩИТА ИНФОРМАЦИИ – предохранение данных и программ от неразрешенного (несанкционированного) использования.

ИГРЫ КОМПЬЮТЕРНЫЕ – большой класс игровых программ для одного или двух партнеров. Нередко вторым партнером выступает сама программа.

ИДЕНТИФИКАТОР – обозначение (наименование) объектов (переменных, массивов, меток и др.) в языках программирования. В последнее еремя для этих же целей чаще применяется болве короткий термин ИМЯ.

ИНТЕРПРЕТАТОР – один из видов программы-переводчика, обеспечивающей пооператорный анализ и псевдоисполнение исходной программы. В результате действия интерпретатора объектный код (машинная программа как эквивалент исходной) не синтезируется.

ИНТЕРФЕЙС – совокупность унифицированных технических и программных средств и правил (соглашений), используемых для сопряжения устройств или программ в вычислительной системе или между системами.

ИНФОРМАТИКА – наука о законах и методах измерения, хранения, переработки и передачи информации с применением математических, программных и технических средств.

ИНФОРМАЦИЯ – совокупность саедений (данных), циркулирующих в обществе, биологических и технических системах. Информация представляется в виде изображений, текстов, звуковых и электрических сигналов и др.

КАНАЛ – совокупность устройств, предназначенных для передачи информации.

КАНАЛ СВЯЗИ – физическая среда, аппаратные и в некоторых каналах программные средства пвредачи информации.

КАТАЛОГ ФАЙЛОВ (ДИРЕКТОРИИ) – информационная структура, содержащая свведения о группе файлов, хранимых совместно на одном носителе.

КИЛОБАЙТ (К) – единица измерения объема памяти ЭВМ, равная 1024 байтвм.

КИЛОБИТ – единица измерения емкости микросхем памяти, равная 1024 битам.

КОБОЛ [от англ. co(mmon) b (usiness) o (riented) l (anguage)] – язык программирования высокого уровня, ориентированный на решение задач экономического характера.

КОД – совокупность знаков (символов) и система правил, при помощи которых информация может быть представлена в виде набора знаков для передачи, обработки и хранения.

КОД МАШИННЫЙ – двоичный код, в котором по специфичным для данного семейства ЭВМ правилам кодируются команды процессора.

КОД ОБЪЕКТНЫЙ – программа, переведенная транслятором с исходного языка (языка программирования) на язык, близкий к машинному коду.

КОДИРУЮЩИЙ ПЛАНШЕТ – см. ДИГИТАЙЗЕР.

КОМАНДА – предписание, инструкция, определяющая шаг выполнения программы. Содержит обозначение операции, адреса (имена) операндов и другие признаки.

КОМАНДА МАШИННАЯ – совокупность элементарных действий, выполняемых ЭВМ по обработке информации, например, сравнение, деление и т.д.

КОМАНД СИСТЕМА – набор всех допустимых для данной ЭВМ, операционной системы или программы команд, предписаний.

КОМАНДНЫЙ ФАЙЛ – средство для предварительного ввода, сохранения и многократного использования последовательности команд диалоговой программы. Широко применяется в системах электронных таблиц, СУБД, в управлении операционными системами.

КОМПИЛЯТОР – программа-транслятор, выполняющая перевод на язык машинных кодов программы, записанной на алгоритмическом языкв. В результате работы компилятора создается объектный код.

КОМПИЛЯТОР ДИАЛОГОВЫЙ – вариант компилятора, допускающий в процессе трансляции общение с пользователем (автором программы).

КОМПИЛЯТОР ОПТИМИЗИРУЮЩИЙ – вариант компилятора, предоставляющий пользователю широкие возможности для отладки программ (например, пошаговое выполнение, визуализация переменных в ходе исполнения, выдача подробных сообщений при любых нарушениях, приводящих к прекращению выполнения программы и др.). Отладочный компилятор синтезирует неэффективный объектный код.

КОМПЬЮТЕР – синоним ЭВМ.

КОМПЬЮТЕР ПЕРСОНАЛЬНЫЙ – небольшая по размерам (настольная) и стоимости (по сравнению с большими ЭВМ) универсальная микроЭВМ, предназначенная для индивидуального использования.

КОНСТАНТА – простое данное (число, строка символов), которое в ходе выполнения программы имеет только одно неизменное значение.

КОНТРОЛЛЕР – устройство управления передачей данных.

КОНФИГУРАЦИЯ ЭВМ – совокупность отдельных блоков и устройств, образующих ЭВМ определенной мощности и назначения. Под базовой конфигурацией понимают минимальный комплект ЭВМ, достаточный для ее нормального функционирования.

КРОСС-КОМПИЛЯТОР – компилятор, осуществляющий перевод программы с исходного языка на некоторой ЭВМ в объектный код ЭВМ другого типа.

КУРСОР – специальный знак (в виде прямоугольника, крестика, стрелки, измененного цветв фона и др.) на экрече дисплея для указания определенных (активных, текущих) позиций или элементов.

ЛЕНТА МАГНИТНАЯ – носитель информации в виде прочной гибкой ленты с нанесенным слоем магнитного вещества. Применяются ленты разной ширины, в кассетах и на катушках. Отличается большой емкостью, дешевизной и низкой средней скоростью доступа к данным.

ЛЕНТА ПЕРФОРИРОВАННАЯ – бумажный носитель информации, запись на который производится однократно путем пробивки (перфорации) отверстий строками, расположенными поперек перфоленты. Каждому символу соответствует своя комбинация отверстий. Наиболее дешевый и наименее удобный носитель, постепенно выходящий из употребления, по крайней мере в вычислительной технике.

ЛИСП – язык программирования для обработки списков.

ЛИСТИНГ – распечатка информации о результатах трансляции, редактирования (связей) и исполнения программы. Содержит исходный текст, таблицы ссылок, диагностические сообщения, результаты выполнения программы. С развитием ЭВМ частота использования листинга как бумажного документа уменьшается.

МАНИПУЛЯТОР "МЫШЬ" – небольшая пластмассовая коробка с несколькими кнопками и кабелем для подсоединения к ЭВМ, предназначенная для управления курсором. Нажатием кнопок инициируют выполнение отмеченых курсором функций. Широко применяется как дополнительное внешнее устройство в персональных компьютерах.

МАРКЕР – 1) специальный знак на носителе информации для обозначения границ, распознавания характера записанных данных и т.д.; 2) то же, что и курсор.

МАССИВ в языке прогрвммирования – совокупность однотипных элементов данных, имеющих одно общее наименование и различающихся в ней порядковым номером.

МАШИНА ВИРТУАЛЬНАЯ – функциональный эквивалент реальной ЭВМ, который моделирувтся при помощи технических средств и специальных служебных программ. Пользователь виртуальной ЭВМ как бы имеет в своем единоличном распоряжении машину с определенными параметрами и решает на ней задачи независимо от других пользователей данной реальной ЭВМ.

МАШИНА ВЫЧИСЛИТЕЛЬНАЯ ОБЩЕГО НАЗНАЧЕНИЯ – ЭВМ для решения широкого круга научно-технических, экономических и других задач. Этим она отличается от специализированной ЭВМ, предназначенной для решения задач одного типа. В данной книге идет речь только о машинах общего назначения.

МЕГАБАЙТ – единица измерения объема памяти ЭВМ, равная 1024 Кбайтам.

МЕНЮ КОМАНДНОЕ – список доступных пользователю команд, функций, действий, появляющийся на экране дисплея для выбора. Один из популярных способов рвализации диалога.

МЕТКА – 1) специальная конструкция в алгоритмическом языке, служащая для обозначения оператора; 2) информация, или этикетка (label), позволяющая операционной системе распознать файл или носитель (том, сменный диск) данных.

МИКРОДИСК МАГНИТНЫЙ ГИБКИЙ (МИКРОДИСКЕТА) – дискета диаметром 89 мм и менее.

МИНИ-ДИСК МАГНИТНЫЙ ГИБКИЙ (МИНИ-ДИСКЕТА, ДИСКЕТА) – магнитный диск диаметром 133 мм.

МИКРОПРОЦЕССОР – арифметическое и логическое устройство, выполненное на полупроводниковых интегральных схемах, часто в виде одной БИС. Используется в качестве главного компонента в составе мини- и микроЭВМ.

МИКРОСХЕМА ИНТЕГРАЛЬНАЯ (ИС) – микроэлектронное изделие в плоском пластмвссовом или керамическом корпусе с металлическими выводами, выполняющее определенную функцию преобразования информации и имеющее высокую плотность упаковки элементов.

МИКРОСХЕМА ИНТЕГРАЛЬНАЯ БОЛЬШОЙ СТЕПЕНИ ИНТЕГРАЦИИ (БИС) – ИС, содержащая от нескольких сотен до нескольких тысяч элементов и компонентов.

МИКРОЭВМ (МИКРОКОМПЬЮТЕР) – ЭВМ, состоящая из микропроцессора, устройств ввода-вывода и запоминания информации, управления. Ориентирована на решение сравнительно простых задач. Отличается малыми габаритами и стоимостью, простотой эксплуатации и обслуживания, высокой надежностью.

МИНИ-ЭВМ – достаточно мощная вычислительная системв, в соствв которой включаются разнообразные ВЗУ и устройства ввода-вывода. Отличаются умеренными характеристиками и сравнительно низкой стоимостью. Нвиболее мощные из мини-ЭВМ последних выпусков превосходят по основным параметрам средние и даже крупные ЭВМ второго и третьего поколений.

МОДЕМ – устройство, состоящее из модулятора и демодулятора. Применяется в качестве адаптера для связи персональных ЭВМ между собой по телефонным линиям.

МОДУЛЬ ИСХОДНЫЙ в программировании – программный модуль, представленный на алгоритмическом языке. Например, подпрограмма на Фортране.

МОДУЛЬ ОБЪЕКТНЫЙ – программный модуль, полученный в результате трансляции исходного модуля.

МОДУЛЬ ПРОГРАММНЫЙ –программа или подпрограмма (процедура), используемая как составная часть других программ либо самостоятельно.

МОДУЛЯТОР – устройство, осуществляющее модуляцию.

МОДУЛЯЦИЯ – изменение параметров некоторого физического процесса (сигнала, переносящего информацию) в соответствии с текущими значениями передаваемого сигнала (модулирующего, представляющего собой кодированную информацию).

МОНИТОР – 1) то же, что дисплей; 2) резидентная программа операционной системы, постоянно присутствующая в ОЗУ и осуществляющая операции управления ЭВМ и обмена с внешними устройствами.

НАКОПИТЕЛЬ НА ГИБКОМ МАГНИТНОМ ДИСКЕ (НГМД) – малогабаритное внешнее запоминающее устройство, в котором в качестве носителя информации используется дискета.

НАКОПИТЕЛЬ НА МАГНИТНЫХ ДИСКАХ (НМД) – внешнее запоминающее устройство прямого доступа с подвижным (вращающимся) запоминающим элементом, в качестве которого применяются сменные и несменные жесткие или гибкие диски.

НАКОПИТЕЛЬ НА МАГНИТНЫХ ДИСКАХ ТИПА "ВИНЧЕСТЕР" (НАКОПИТЕЛЬ НА ЖЕСТКИХ МАГНИТНЫХ ДИСКАХ) – см. ВИНЧЕСТЕРСКИЙ НАКОПИТЕЛЬ.

НАКОПИТЕЛЬ НА МАГНИТНОЙ ЛЕНТЕ (НМЛ) – внешнее запоминающее устройство последовательного доступа с подвижным носителем – магнитной лентой. Часто для краткости называют магнитофоном, в простейших ЭВМ в качестве НМЛ нередко используют бытовые магнитофоны.

НАКОПИТЕЛЬ НА СМЕННОМ ОПТИЧЕСКОМ ДИСКЕ (НСОД) – устройство внешней памяти со сменным опто-магнитным диском многократной записи. Габариты НСОД фирмы Canon раааны полновысотному 133 мм НГМД, а размеры самого оптического диска емкостью 256 Мбайт близки к размерам мини-дискеты. Впервые применен в качестве основного внешнего ЗУ в настольной ПЭВМ NeXT.

НОСИТЕЛЬ ДАННЫХ (ИНФОРМАЦИИ) – устройство или физическая среда для хранения в определенной форме различной информации. Основные характеристики носителей: плотность записи и суммарный объем, долговечность, надежность и простота записи-считывания, возможность многократной записи.

ОБЕСПЕЧЕНИЕ АППАРАТНОЕ – соаокупность технических компонентоа ЭВМ или вычислитвльной системы.

ОБЕСПЕЧЕНИЕ МАТЕМАТИЧЕСКОЕ – комплекс программных средста, реализующих различные вычислительные методы. Прежде математическим обеспечением называли вообще любые программы для ЭВМ.

ОБЕСПЕЧЕНИЕ ПРОГРАММНОЕ – соаокупность программ общего применения – операционные системы, трансляторы, библиотеки математических подпрограмм, СУБД и многие другие программы и их комплексы, поставляемые как в составе ЭВМ, так и независимо.

ОКНО – формируемая на экране дисплея прямоугольная рамка, в которую помещено иное по сравнению с окружающим фоном содержимое.

ОПЕРАТИВНОЕ ЗАПОМИНАЮЩЕЕ УСТРОЙСТВО (ОЗУ) – см. ЗАПОМИНАЮЩЕЕ УСТРОЙСТВО.

ОПЕРАТИВНАЯ ПАМЯТЬ – то же, что ОЗУ.

ОПЕРАНД – элемент данных, над которым выполняется операция.

ОПЕРАЦИОННАЯ СИСТЕМА (ОС) – см. СИСТЕМА ОПЕРАЦИОННАЯ.

ОПЕРАЦИЯ ВВОДА-ВЫВОДА – действия по обмену информацией между ОЗУ и одним из устройств ввода-вывода.

ОПЕРАЦИЯ МАШИННАЯ – совокупность действий, выполняемых одной командой ЭВМ.

ОТЛАДКА – процесс выявления ошибок и исправления программы как один из этапов ее разработки. Для облегчения и ускорения отладки пользуются специальными отладочными приемами и средстааами.

ОТЛАДЧИК – вспомогательная программа для отладки других программ. Обеспечивает разные режимы исполнения отлаживаемой программы, выдает диагностическую информацию и промежуточные результаты.

ПАКЕТ ДИСКОВ – совокупность магнитных дисков, имеющих общий вал. Используется в качестве сменного или постоянного носителя информации.

ПАМЯТИ ЗАЩИТА – соаокупность аппаратных и программных средста в ЭВМ, обеспечивающая независимость данных одной задачи от возможного разрушающего алияния другой при многозадачной обработке информации.

ПАМЯТЬ ВНЕШНЯЯ – память ЭВМ, реализуемая ЗУ на магнитных дисках, лентах, барабанах, интегральных схемах и др.

ПАМЯТЬ ВНУТРЕННЯЯ – память ЭВМ, конструктиано объединенная с центральным устройстаом (процессором) и осуществляющая хранение команд и данных, необходимых для выполнения программы.

ПАМЯТЬ ОПЕРАТИВНАЯ – основная часть внутренней памяти ЭВМ, реализуемая ОЗУ и предназначенная для временного хранения программы и данных в процессе аыполнения арифметических и логических операций. Основные характеристики – емкость и цикл обращения (быстродействие).

ПАМЯТЬ СВЕРХОПЕРАТИВНАЯ – см. ЗАПОМИНАЮЩЕЕ УСТРОЙСТВО СВЕРХОПЕРАТИВНОЕ.

ПАСКАЛЬ – язык программироаания высокого уровня со строгим и простым синтаксисом, предложенный его создатвлем Н. Виртом пераоначально для целей обучения программироаанию, а впоследстаии нашедший применение как язык системного и прикладного назначения.

ПЕРЕДАЧА УПРАВЛЕНИЯ – отклонение от прямой последовательности аыполнения команд в ЭВМ. В языках программирования она эквивалентна переходу по метке (оператор GOTO) и входит неявным элементом в ряд других конструкций (цикл, условный оператор и др.).

ПИКСЕЛ [pixel] – минимальный элемент изображения на экране, состоящий из нескольких [цветных] точек и рассматриааемый в программе как одна точка определенной яркости или оттенка.

ПЛ/1 – язык программироаания аысокого уровня, предназначенный для решения широкого круга экономических и научно-технических задач. Отличается избыточностью и недостаточной строгостью синтаксиса, что затрудняет его изучение и освоение, а также препятствует созданию надежных программ.

ПОДСКАЗКА – см. ПОМОЩЬ.

ПОДПРОГРАММА – оформленная по правилам языка часть программы, имеющая самостоятельное значение и допускающая обращение к ней из различных точек программы. Имеет имя (идентификатор), перечень параметров и их особенностей и составляющие тело подпрограммы операторы (языковые конструкции).

ПОДПРОГРАММА СТАНДАРТНАЯ – подпрограмма, состааленная таким образом, чтобы ее легко могли использовать другие лица при решении задач определенного класса. Существуют обширные библиотеки стандартных математических подпрограмм, которые применяются при решении всеаозможных научно-технических задач.

ПОЛЬЗОВАТЕЛЬ – лицо, использующее данное вычислительное устройство или программное средство для выполнения необходимых ему работ.

ПОМОЩЬ [англ. help] – встроенные справочные средства, дающие пользователю по его запросу минимальные инструкции.

ПРИНТЕР – обобщенное наименование устройств печати разных типов – барабанных, мозаичных, лазерных и др.

ПРИНЦИП УМОЛЧАНИЯ – принятый во многих языках программирования и операционных системах принцип неявного определения свойств, приписываемых объектам языка, командам или параметрам системы. Если некоторому свойству можно задать одно из нескольких значений, то наиболее вероятное из них присваивается автоматически. В ряде случаев "умалчиваемое" свойство назначается в зависимости от контекста, в котором употребляется конструкция или команда.

ПРОГРАММ БИБЛИОТЕКА – совокупность программ, предназначенных для многократного использования, вместе с системой, обеспечивающей их обозначение, хранение и применение.

ПРИНЦИП WYSIWYG [What You See Is What You Get] – "что вижу, то и получаю" – принцип устройства человеко-машинного интерфейса, при котором видимое на экране дисплея изображение создаваемого документа точно соответствует получаемой на бумаге копии. Выполнение этого принципа обязательно, например, в издательских системах. Произносится "визивиг".

ПРОГРАММА – запись алгоритма решения задачи на языке программирования высокого уровня или в машинных кодах.

ПРОГРАММА-ДИСПЕТЧЕР – см. МОНИТОР (2).

ПРОГРАММА ИСХОДНАЯ – обычно имеется в виду программа, составленная на языке высокого уровня или автокоде.

ПРОГРАММА ОБЪЕКТНАЯ – см. КОД ОБЪЕКТНЫЙ.

ПРОГРАММА ПРИКЛАДНАЯ – программа, реализующая решение конкретной прикладной задачи.

ПРОГРАММА РАБОЧАЯ – программа на языке машинных команд, полностью готовая к немедленному исполнению, обычно получается из исходной трансляцией и редактированием связей (сборкой).

ПРОГРАММА-РЕДАКТОР – см. РЕДАКТОР СВЯЗЕЙ и РЕДАКТОР ТЕКСТОВ.

ПРОГРАММА СИСТЕМНАЯ – нестрогое общее название программ, относящихся к операционным системам, инструментарию и всевозможному сервису ("все, что не прикладное").

ПРОГРАММ ПРИКЛАДНЫХ ГЕНЕРАТОР – см. ГЕНЕРАТОР ПРИКЛАДНЫХ ПРОГРАММ.

ПРОГРАММИРОВАНИЕ для ЭВМ – составление программ решения задач, включающее в себя разработку алгоритма, запись его на языке программирования, перевод на машинный язык, поиск и устранение ошибок, тестовый прогон для подтверждения работоспособности.

ПРОЦЕДУРА – название подпрограммы в некоторых языках (Алголе, Паскале и др.)

ПРОЛОГ [pro(gramming in) log(ic)] – язык для разработки программ в области искусственного интеллекта, служит средством реализации экспертных систем (которые перерабатывают правила и факты в "человеческой манере"), интерфейса на естественном языке, управления данными, моделирования и других приложений, используемых при распознавании образов и логическом выводе.

ПРОЦЕССОР – 1) центральное устройство ЭВМ, которое выполняет арифметические и логические операции, заданные программой, управляет вычислительным процессом и координирует работу периферийных устройств. Наличие нескольких процессоров, возможно разнотипных, ускоряет выполнение программ. Существует много процессоров других типов – ввода-вывода, математический, коммуникационный, дисплейный и др; 2) сложная большая программа, предназначенная, например, для работы с текстами (word processor).

ПСЕВДОКОД – способ описания логики программы на частично формализованном естественном языке. Позволяет "крупными мазками" формулировать основные идеи алгоритма перед точной его записью на языке программирования.

РАСПЕЧАТКА – см. ЛИСТИНГ.

РАФОС – одна из популярных операционных систем, используемых на СМ ЭВМ. Аналог зарубежной ОС RT-11.

РЕГИСТР – элемент сверхоперативной памяти, аналог ячейки увеличенного быстродействия, предназначенный для приема и хранения операнда (числа, команды, адреса и пр.), а также выполнения определенных операций над ним. Разрядность регистра обычно совпадает с разрядностью ячейки ОЗУ либо кратна ей.

РЕДАКТОР СВЯЗЕЙ – обслуживающая программа для объединения нескольких отдельно переведенных на машинный язык (объектных) модулей в одну рабочую программу. Иногда называется сборщиком.

РЕДАКТОР ТЕКСТОВ – обслуживающая программа для подготовки текстов программ, документов, описаний. Существует множество вариантов на всех типах ЭВМ. Наиболее развитые многофункциональные редакторы применяются в редакционно-издательском деле ("настольных" издательствах).

РЕЖИМ ДИАЛОГОВЫЙ – см. ДИАЛОГОВЫЙ РЕЖИМ.

РЕЖИМ РЕАЛЬНОГО ВРЕМЕНИ – режим обработки данных, при котором ЭВМ взаимодействует с внешними по отношению к ней процессами (измеряемыми и управляемыми) в темпе, сопоставимом со скоростью их протекания.

СБОЙ – кратковременная утрата работоспособности элементов и блоков ЭВМ. Часто после сбоя работоспособность восстанавливается автоматически, но содержимое ОЗУ, как правило, портится.

СЕТЬ МИКРОЭВМ ЛОКАЛЬНАЯ – совокупность нескольких микроЭВМ, сосрвдоточенных на ограниченной территории (на расстоянии сотен метров, если мерять по соединительным линиям), саязанных между собой специально созданной кабельной сетью.

СЕТЬ ТЕРМИНАЛЬНАЯ – совокупность нескольких терминалов, расположенных на ограниченной территории и подключенных через адаптер (терминальную станцию) к достаточно мощной ЭВМ, способной обслуживать одновременно нескольких пользователей.

СЕТЬ ЭВМ – совокупность ЭВМ, распределенных по некоторой территории и связанных между собой каналами передачи информации.

СИ – язык программирования высокого уровня, наделенный возможностями автокода в части доступа к содержимому ячейки и машинным командам. Широко применяется в системном программировании.

СИМВОЛ – отдельный знак (буква, цифра, скобка, знак арифметической операции и др.) из заданного набора, применяемого для представления данных в ЭВМ, записи программ, кодирования информации.

СИСТЕМА ВЫЧИСЛИТЕЛЬНАЯ – автоматизированная система обработки данных, представляющая собой набор средств вычислительной техники, периферийного оборудования и программного обеспечения для приема, хранения, переработки и выдачи результатов пользователям в наиболее удобной для них форме.

СИСТЕМА ОПЕРАЦИОННАЯ (ОС) – комплекс программ, обеспечивающий нормальное функционирование ЭВМ и эффективное использование ее ресурсов. При отсутствии ОС любая современная универсальная ЭВМ (за исключением простейшей) практически неработоспособна. Основными компонентами ОС являются резидвнтные управляющие программы, редакторы, загрузчики, трансляторы, средства генерации, программы обслуживания.

СИСТЕМА МАЛЫХ ЭВМ (СМ ЭВМ) – семейство программно-совместимых машин малой производительности, используемых в системах управления технологическими процессами, автоматизации научных исследований, проектирования, управления промышленными объектами, проведения расчетов, обучения.

Программное обеспечение СМ ЭВМ построено по модульному принципу, что дает возможность компоноаки программных средств в соответствии с требуемыми режимами работы и выполняемыми функциями. В состав программного обеспечения входят ОС различного назначения, системы программирования, сервисные и контрольно-диагностические программы. Популярными представителями семейств являются СМ-3, СМ-4, СМ-1420, ДВК-1,2,3,4, "Электроника-60", БК 0010.

СИСТЕМА НАСТОЛЬНАЯ ИЗДАТЕЛЬСКАЯ – совокупность технических и программных средств, обеспечивающая первоначальный набор (ввод), верстку с подбором шрифтов, подготовку иллюстративного материала и печать малотиражных изданий или выдачу оригиналов-маквтов для размножения большим тиражом в типографии. Основу издательской системы состааляет персональная ЭВМ, снабженная совокупностью программных средств для полной подготовки издания. Дополнительно может включать в себя лазерный и другие современные принтеры для получения высококачественных оттисков. При небольшом тираже осуществляет весь процесс вплоть до брошюровки готового издания. Наилучшее полиграфическое качество достигается при включении в состав системы фотонаборной машины.

СИСТЕМА ОПЕРАЦИОННАЯ ДИСКОВАЯ (ДОС) – тип ОС, использующих в качестве основной внешней памяти запоминающее устройство на магнитных дисках. Большинство компонентов ДОС при работе ЭВМ размещается на дисках, за исключением резидентной части, постоянно находящейся в ОЗУ, что обеспечивает в целом высокое быстродействие машины.

СИСТЕМА ПРОГРАММИРОВАНИЯ – комплекс языковых и программных средств, предназначенных для автоматизации процесса составления, отладки программ и подготовки их к выполнению. Включает в себя язык программирования, один или несколько трансляторов разных типов, библиотеку подпрограмм, языково-ориентированный текстовый редактор, отладчик. Часть из этих компонентов может отсутствовать.

СКАНЕР – внешнее (периферийное) устройство, предназначенное для ввода изображений или текстов без использования клавиатуры.

СЛОВО МАШИННОЕ – соаокупность битов определенного объема, постоянного для ЭВМ данного типа и равного размеру ячейки памяти. Хранится и воспринимается при обработке как единая порция информации, в частности, команда. Длина машинного слова колеблется от 8 в простейших до 128 бит в наиболее мощных ЭВМ.

СОВМЕСТИМОСТЬ ЭВМ ПРОГРАММНАЯ – возможность выполнения одних и тех же программ на разных ЭВМ с получением идентичных результатов.

СОПРОВОЖДЕНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ – развитие сущестaующего ПО его авторами с целью устранения ошибок, выявленных в ходе эксплуатации, адаптации его к новым требованиям пользователя по мере накопления опыта и расширения сфер применения ПО. Полученный в процессе сопровождения материал служит базой для новых разработок программных средств.

СТЕК – оперативная память, организованная бвз адресов, по принципу оружейного магазина для патронов. Делится на ячейки одинакового размвра, имеет вершину (вход), через которую загружается кодами операций и значениями операндов. Применяется при инввbooрсной записи алгоритмов.

СТРИМЕР [от stream] – накопитель на магнитной ленте особой конструкции, предназначенный для скоростного копирования всего содержимого накопителя на жестких магнитных дисках. Применяется для страховочного дублирования информации. Скоростные и емкостныв параметры стримера согласованы с характеристиками НЖМД, и копирование выполняется в так называемом режиме "бегущей ленты", что намного быстрее, чем в обычных НМЛ.

СТРУКТУРА БЛОЧНАЯ ПРОГРАММЫ – организация программы в виде совокупности смежных или вложенных блоков.

СУБД – см. БАЗА ДАННЫХ.

СУПЕРВИЗОР – см. МОНИТОР (2).

СУПЕР-ЭВМ – ЭВМ производительностью свыше 100 млн. операций в секунду, имеющие оперативную память в сотни и тысячи мегабайт и оперирующие словами, длиной не менее 64 разрядов.

СУПЕРМИКРОЭВМ – микроЭВМ с производительностью свыша 1 млн. операций в секунду, оперативной памятью в несколько Мбайт и длиной слова не менее 32 разрядов. Персональные ЭВМ на базе микропроцессора Intel 80386 (32-разрядные) – типичные представители класса супермикроЭВМ.

СУПЕРМИНИ-ЭВМ – мини-ЭВМ, по структуре, стоимости и габаритным параметрам близкая к обычным мини-ЭВМ, а по производительности (свыше 1 млн. операций в секунду) и емкости ОЗУ (десятки Мбайт) – к большим ЭВМ.

СХЕМА ИНТЕГРАЛЬНАЯ – см. МИКРОСХЕМА ИНТЕГРАЛЬНАЯ.

ТАБЛИЦА ЭЛЕКТРОННАЯ (КРУПНОФОРМАТНЫЙ БЛАНК, spreadsheets) способ наглядного представления данных на экране дисплея и организации срвдств для их обработки в виде специальной программы. Электронные таблицы широко применяются в экономической сфере и для научно-техничвских расчетов.

ТАЙМЕР – аппаратно-программные средства в вычислительной машине, выполняющие функции календаря и часов астрономического времени. Во многих моделях персональных ЭВМ таймер питается от батарейки и не зависит от работы отдельных устройств.

ТЕРМИНАЛ – оконвчный пункт вычислительной системы или свти связи, используемый для ввода и вывода данных. Основой терминвла чаще всего служит комплект из дисплея и клавиатуры. По возможностям обработки информации различают интеллектуальные и нвинтеллектуальные терминалы, по способу подключения – удаленные и локальные.

ТРАНСПЬЮТЕР – несамостоятельная ЭВМ минимальной конфигурации, состоящая из процесора, оперативной памяти и средств сопряжения с ведущей ЭВМ. Берет на себя выполнение части вычислительных операций, позволяя тем самым повысить суммарное быстродействие машины. Соединением нескольких (иногда сотен и даже тысяч) транспьютеров в одну систему добиваются производительности, характерной для суперЭВМ. Эффективное использование транспьютеров требует адаптации базовой операционной системы, трансляторов и программ. Транспьютеры выпускаются, в частности, фирмой Microway в виде сменной платы к ПЭВМ семейства IBM PC на основе процессора 80386 в одинарном (monoputer) и счетверенном (quadputer) вариантах.

ТРАНСЛЯТОР – программа, предназначенная для перевода (трансляции) описаний алгоритмов с одного (исходного) языка на другой (результирующий, выходной). Наиболее распространены трансляторы с языков высокого уровня на машинные и машинно-ориентированные языки.

ТУРБО – режим работы ПЭВМ, отличающийся от базового (основного) повышенной на несколько мегагерц тактовой частотой. В этом режиме могут неправильно работать некоторые программы, рассчитанные на невысокое быстродействие.

ТУРБОСИСТЕМА – интегрированная система для ввода и отладки программ, в которой с целью значительного (на порядок и более) уменьшения времени реакции ЭВМ собраны в единое целое текстовый редактор, транслятор, редактор связей и загрузчик. Взаимодействие с пользователем осуществляется в форме меню-диалога. Турбосистема может накладывать на язык и программу некоторые ограничения. Богатые по функциональным возможностям "турбо" требуют увеличенного объема ОЗУ (150...300 К). Широкое распространение на ПЭВМ получили системы Турбо-Паскаль, Турбо-Си, Турбо-Пролог. Заметим, что режим работы ПЭВМ "турбо" и турбосистемы объединяет лишь то, что они обозначают нечто более быстрое по сравнению с устоявшимся, общепринятым.

УМОЛЧАНИЯ ПРИНЦИП – см. ПРИНЦИП УМОЛЧАНИЯ.

УСТРОЙСТВА ВНЕШНИЕ (ПЕРИФЕРИЙНЫЕ) – устройства для внешнего (по отношению к процессору и оперативной памяти) преобразования и хранения информации. К ним относятся также всевозможные устройства ввода-вывода.

УСТРОЙСТВО АРИФМЕТИКО-ЛОГИЧЕСКОЕ (АЛУ) – центральная часть процессора, служащая для выполнения арифмвтических и логических операций над данными, поступающими из ОЗУ. АЛУ соаременных ЭВМ могут выполнять (в зависимости от архитектуры) нвсколько десятков или сотен различных операций (команд).

УСТРОЙСТВО ПЕЧАТАЮЩЕЕ АЛФАВИТНО-ЦИФРОВОЕ – см. АЦПУ.

УСТРОЙСТВО ПОДГОТОВКИ ДАННЫХ – периферийное независимое от ЭВМ устройство, предназнвченное для фиксации данных на носителях информации (перфокартах, магнитных лентах и дисках) с целью последующего ввода их в ЭВМ.

УСТРОЙСТВО РЕЧЕВОГО ВВОДА – см. ВВОД РЕЧЕВОЙ.

УСТРОЙСТВО РЕЧЕВОГО ВЫВОДА – см. ВЫВОД РЕЧЕВОЙ.

УТИЛИТЫ – вспомогательные автономные системные программы в составе многих ОС, обеспечивающие пользователя различным сервисом.

ФАЙЛ – последовательность записей, размещаемая на анешних ЗУ и рассматриваемая при хранении и обработке как единое целое. Обычно файлы обозначаются алфавитно-цифровыми именами.

ФАЙЛ КОМАНДНЫЙ – объединенная в виде файла последовательность командных строк, исполняемая операционной системой или иной диалоговой программой как последовательность команд пользователя, выданных им с терминала или клавиатуры ПЭВМ.

ФЛОППИ-ДИСК – см. ДИСКЕТА.

ФОРТ [FORTH] – язык программирования аысокого уровня с простым, но своеобразным синтаксисом, занимающий промежуточное положение между языками трансляции (такими, как Алгол, Фортран, Паскаль) и интерпретации (Бэйсик, АПЛ). Арифметические выражвния в Форте записываются в так назыаавмом инверсном виде, основной особенностью которого яаляется то, что знак операции записывается после соответстаующих операндов, а не мвжду или перед ними, как в большинстве известных языков.

ФОРТРАН [от англ. for (mula) tran (slation)] – язык программирования высокого уровня, предназначенный для формализованной записи алгоритмов решения научно-технических и инженерных задач. Наиболее распространен вариант Фортран-4, сложившийся окончательно в 60-е годы. В последнее время ему на смену пришел Фортран-77, который отличается многочисленными улучшениями, приблизившими этот язык к современным аысоконадежным языкам структурного программирования. Практически на всех типах ЭВМ имеются трансляторы для того или иного варианта Фортрана.

ФУНКЦИЯ ВСТРОЕННАЯ – описанная в языке или программной системе математичвская, логическая или иная функция, предусмотренная разработчиками для удобства пользователей. Математический алгоритм, реализующий функцию, помещен в библиотеку транслятора (программную систему) и скрыт от пользователя. Для обращения к встроенной функции достаточно указать ее имя и, если требуется, аргументы (операнды). Например, набор встроенных тригонометрических функций (sin, cos и др.) ао многих языках.

ЧИП – интегральная микросхема в отдельном корпусе.

ЭЛЕКТРОННАЯ ВЫЧИСЛИТЕЛЬНАЯ МАШИНА (ЭВМ) – вычислительная машина, в которой основныв функциональные элементы (логические, запоминающие, индикационные и т.д.) выполнены на злектронных приборах. Различают аналоговые (АВМ) и цифровые (ЦВМ) ЭВМ, причем в последние годы под термином ЭВМ понимаются именно цифровые (если нв огоеорено иное) вычислительные машины. В данной книге речь идет только о цифровых ЭВМ.

В настоящее время в мире выпускаются ЭВМ в основном четвертого поколения на больших интегральных схемах и разрабатываются ЭВМ пятого поколения. В состав ЭВМ аходят один или несколько процессоров, основная (оперативная – ОЗУ) память, внешние запоминающие устройства, каналы и устройстаа ввода-вывода. По назначению ЭВМ делятся на универсальные (общего назначения) и специализированные. По конструкции и производительности различвют микро- и мини-ЭВМ, малые, средние, большие и супвр-ЭВМ. Любая обработка информации в ЭВМ осуществляется по заранее составленной программе путем выполнения совокупности входящих в нее машинных команд. Ввод и вывод программ и данных осуществляются с помощью периферийных устройств. ЭВМ, особенно пврсональные, могут применяться всюду, где имеют место возникновение, передача, хранение и какое-нибудь преобразование информации любого вида и содержания.

ЭМУЛЯТОР – аппаратное или программное средство, которое на данной ЭВМ исполняет (модвлирует, имитирует, эмулирует) команды другой ЭВМ (обычно в замедленном темпе) и позволяет тем самым выполнять программы, предназначенные для "чужих" машин. Эмуляция с помощью программ используется, в частности, при разработке машин новых архитектур.

ЯЗЫК АЛГОРИТМИЧЕСКИЙ – совокупность синтаксичвски связанных алфавита (множвства допустимых символов), тврминов (служебных слов) и конструкций, позволяющая по опрвделвнным правилам описывать алгоритмы рвшения задач. С алгоритмичвского языка текст программы пвреводится транслятором на язык конкретной ЭВМ или исполнявтся ею посредством интврпретатора.

ЯЗЫК ВЫСОКОГО УРОВНЯ – язык программирования, средства которого допускают запись алгоритма в наглядном, лвгко восприниманмом виде. Программа на языкв высокого уровня может исполняться на ЭВМ с любой системой команд при наличии транслятора (интерпретатора).

ЯЗЫК МАШИННО-ОРИЕНТИРОВАННЫЙ – язык программирования, который по типам данных и синтаксической структурв отражавт структуру и систвму команд ЭВМ, но вместв с твм обладавт свойствами, упрощающими и автоматизирующими процесс программирования. Такой язык позволяет писать программы, нв уступающие по эффвктивности программам, написанным нвпосредствванно в командах машины. Используется преимущвственно в систвмном программировании. Типичный представитель – язык ассемблвр.

ЯЗЫК МАШИННЫЙ – формальный язык для записи послвдовательности команд решения задачи, содержанив и правила которого реализуются непосредственно аппаратурой конкрвтной ЭВМ. Иногда машинным языком называют также систему команд ЭВМ.

ЯЗЫК НИЗКОГО УРОВНЯ – язык программирования, отличающийся большой двтализацией шагов при формулировкв задания для ЭВМ, он (автокод, ассемблер) близок к машинному языку соответствующвй ЭВМ. Программа на языке низкого уровня, как правило, машинно-зависима и нв может выполняться на ЭВМ с иной систвмой команд.

ЯЗЫК ПРОГРАММИРОВАНИЯ – формальный язык, применяемый для описания информации (данных) и алгоритма (программы) их обработки на ЭВМ. Часто на практике не делают различия мвжду алгоритмичвским языком и языком программирования высокого уровня, среди которых наиболее распространены Алгол, Бейсик (Бэйсик), Кобол, Модупа, Паскаль, ПЛ/1, Пролог, Си, Фортран и другие.

ЯЗЫК ПРОЕКТИРОВАНИЯ ПРОГРАММ – некомпилируемый язык, предназначенный для описания структур программ и алгоритмов. В некоторых новейших разработках (например, в систвме Layout) предпринята попытка увязать проектированив с программированием и обвспвчить трансляцию с языка провктирования на язык програмирования.

ЯЗЫКИ ПРОГРАММИРОВАНИЯ РАСШИРЯЕМЫЕ – языки, содвржащив срвдства определения новых языковых конструкций, уввличивающих их мощность. Примвненив расширявмого языка позволявт с помощью одного и того же транслятора осуществлять перевод с различных вариантов языка, твм самым приспосабливая его к разным классам задач без дополнитвльных затрат на реализацию (изготовлвние новых трансляторов). Характерный представитвль семейства расширявмых языков – Алгол-68.

ЛИТЕРАТУРА

1. **Абрамов В.В., Розенталь Ю.Д.** Внешние запоминающие устройства для персональных ЭВМ//ЭВМ массового применения. – М.: Наука, 1987.– С. 184–189.
2. **Задачи** по программированию/С.А.Абрамов, Г.Г.Гнездилова, Е.Н.Капустина, М.И.Селюн. – М.: Наука, 1988. – 224 с.
3. **Баранов С.Н., Ноздрунов Н.Р.** Язык Форт и его реализации.– Л.: Машиностроение. Ленингр. отделение, 1988. – 157 с.
4. **Брябрин В.М.** Программное обеспечение персональных ЭВМ. – М.: Наука, 1988. – 272 с.
5. **В мире** персональных компьютеров. – М.: IDG Communication; Радио и связь, 1988. – № 1,2; Мир ПК. – М.: IDG Communication, 1989. – № 3,4,5.
6. **ГОСТ** 27201-87. Типы, основные параметры, общие технические требования к персональным ЭВМ.
7. **Готье Р.** Руководство по операционной системе UNIX. – М.: Финансы и статистика, 1985. – 232 с.
8. **Дробушевич Г.А.** Словарь программиста. – Минск: Вышэйшая школа, 1988. – 143 с.
9. **Заморин А.П., Мячев А.А., Селиванов Ю.П.** Вычислительные машины, системы, комплексы. Справочник. – М.: Энергоатомиздат, 1985.–264 с.
10. **Иоффе А.Ф.** Персональные ЭВМ в организационном управлении. – М.: Наука, 1988. – 208 с.
11. **Клоксин У., Меллиш К.** Программирование на языке Пролог. – М.: Мир, 1987. – 336 с.
12. **Котов Ю.В.** Как рисует машина. – М.: Наука, 1988. – 224 с.
13. **Лопато Г.П., Смирнов Г.Д., Пыхтин В.Я.** Советские персональные ЭВМ Единой системы//ВТ соц.стран. – М.: Финансы и статистика, 1986. Вып. 20. С. 3–12.
14. **Майоров С.А., Кириллов В.В., Приблуда А.А.** Введение в микроЭВМ. – Л.: Машиностроение. Ленингр. отделение, 1988. – 304 с.
15. **МикроЭВМ:** В 8 кн. : Кн. 5. Персонально-профессиональные ЭВМ/Под ред. Л.Н.Преснухина.– М.: Высшая школа, 1988. – 143 с.
16. **Микропроцессоры:** В 3 кн. : Кн 1. Архитектура и проектирование микроЭВМ. Организация вычислительных процессов/П.В.Нестеров, В.Ф.Шаньгин, В.Л.Горбунов и др.; Под ред. Л.Н.Преснухина. – Минск: Вышэйшая школа, 1987. – 414 с.
17. **Надььердь И., Кевари И.** Развитие лазерныу периферийных устройств//ВТ соц.стран. – М.: Финансы и статистика, 1987. Вып. 22. С. 36–44.
18. **Персональный** компьютер: рабочее место профессионала.– М.: Наука, 1989. – 172 с.
19. **Черемных С.В., Гиглавый А.В., Поляк Ю.Е.** От микропроцессоров к персональным ЭВМ. – М: Радио и связь, 1988. – 288 с.
20. **Crall R.K.** What's NeXT? – Computers in Physics, 1989. – V. 3, N. 5, pp. 93–100.

Научно-популярное издание

Массовая радиобиблиотека. Вып. 1163

ПОПКОВ Анатолий Иванович

ВВЕДЕНИЕ В ПРАКТИЧЕСКУЮ ИНФОРМАТИКУ

Руководитель отдела А.В.Соснин
Редактор Н.М.Шпагина
Обложка художника В.И.Моськпана
Корректор Э.Г.Пелымская

ИБ № 2175

Подписано в печать с оригинал-макета 11.04.90 КЗ 06030
Формат 84x108 1/32. Бумага офсетная. Гарнитура "Цюрих".
Усл. печ. л. 8,4. Усл. кр.-отт. 8,65. Уч.-изд. л. 10,2 Тираж 200000 экз.
Изд. № 028 Заказ № 172 Цена 1 р 10 к.
Томское отделение издательства "Радио и связь"
634055, Томск, а/я 2211.

Отпечатано в 4-й типографии издательства "Наука".
630077, Новосибирск, Станиславского, 25.

Уважаемый читатель!

Томское отделение издательства "Радио и связь" в 1990 году выпускает книгу из серии "Массовая радиобиблиотека":

Поддубная Т.Н., Фукс И.Л. Информатика в задачах и упражнениях. Объем 10 л. Цена 80 к.

Рассматривается ряд задач, связанных с моделированием на ЭВМ реальных объектоа и организацией различных способов обработки числовой и символьный информации. Решение задач доведено до уровня готовых алгоритмов и программ и сопровождается разнообразными упражнениями и заданиями. Подбор задач и характер изложения направлены на иллюстрацию основных понятий и методов информатики.

Для преподавателей информатики, школьников и студентов, изучающих этот предмет.

Томское отделение издательства "Радио и связь" готовит к изданию в 1991 году книгу из серии "Массовая радиобиблиотека":

Рыжов В.А., Волков В.Г., Шмырин Г.П. Эквалайзеры. 10 л. Цена 80 к.

Рассмотрены требования, возникающие при разработке устройств частотной коррекции звуковых трактов воспроизведения – эквалайзеров. Приведены методы проектирования эквалайзеров, эквалайзеров-анализаторов с ручным и автоматическим управлением. Рассмотрены схемотехнические решения конкретных типов эквалайзеров.

Для широкого круга радиолюбителей, может быть полезна руководителям радиокружков.